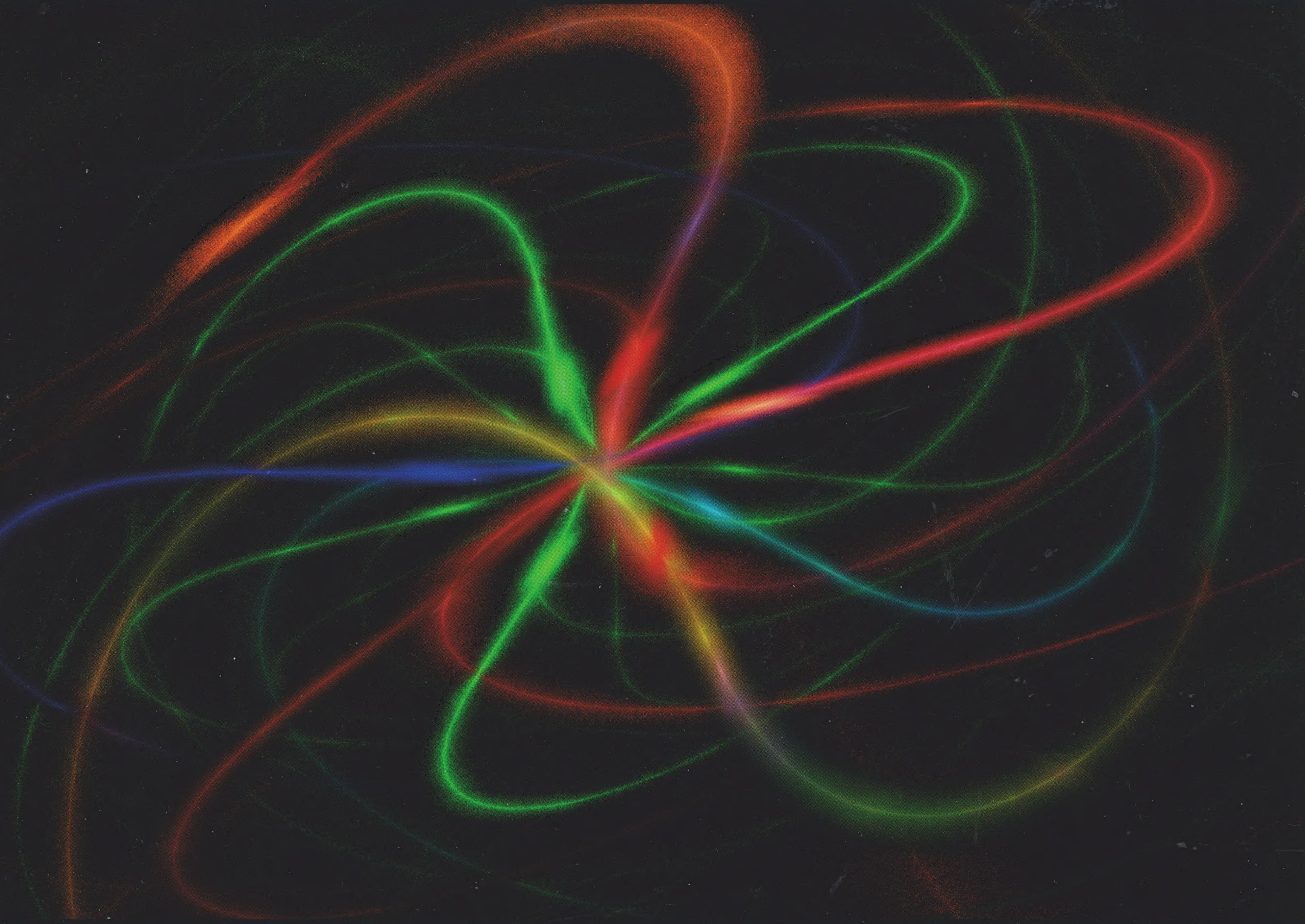
MST125

Essential mathematics 2

The Open University



MST125 Computer algebra guide

The Open University

MST125

Essential mathematics 2

Computer algebra guide

This publication forms part of an Open University module. Details of this and other Open University modules can be obtained from Student Recruitment, The Open University, PO Box 197, Milton Keynes MK7 6BJ, United Kingdom (tel. +44 (0)300 303 5303; email general-enquiries@open.ac.uk).

Alternatively, you may visit the Open University website at www.open.ac.uk where you can learn more about the wide range of modules and packs offered at all levels by The Open University.

The Open University, Walton Hall, Milton Keynes, MK7 6AA.

First published 2015. Second edition 2019.

Edited, designed and typeset by The Open University, using LaTeX.

Printed in the United Kingdom by Page Bros, Norwich.

FSC
www.fsc.org
MIX
Paper from
responsible sources
FSC® C023114

SUP 06241 0

2.1

Contents

Introduction **5**

1 Installing, testing and configuring Maxima **7**

1.1 Introducing and installing Maxima 7

1.2 Starting, testing and configuring wxMaxima 9

2 Getting started with Maxima **14**

2.1 Calculating with Maxima 14

2.2 wxMaxima cell markers 18

2.3 Decimal numbers 19

2.4 Troubleshooting wxMaxima 20

2.5 Reusing and editing commands 21

2.6 Variables 23

2.7 System variables 26

2.8 Annotating your wxMaxima worksheet 27

2.9 Saving and printing your work 28

2.10 Getting help 31

3 Using Maxima **33**

3.1 Algebra and functions 33

3.2 Plotting graphs 38

3.3 Solving equations 46

3.4 Calculus 51

4 Number theory **58**

5 Parametric equations **60**

5.1 Plotting curves from parametric equations 60

5.2 Plotting conics parametrically 63

6 Partial fractions and polynomial division **64**

6.1 Partial fractions 65

6.2 Polynomial division 66

7 Differential equations **67**

7.1 Solving differential equations analytically 67

7.2 Plotting direction fields 71

7.3 Solving differential equations numerically 76

8 Eigenvalues and eigenvectors **79**

8.1 Matrices 79

8.2 Characteristic equations, eigenvalues and eigenvectors 82

9 Solving linear recurrence systems **86**

10 Maxima accessibility guide **90**

10.1 Choosing and configuring your interface 90

10.2 Starting, testing and configuring Maxima using the command-line interface 92

10.3 Troubleshooting Maxima using the command-line interface 96

10.4 Reusing and editing commands in the command-line interface 96

10.5 Saving and printing your work using the command-line interface 97

10.6 Plotting graphs without wxMaxima 101

10.7 Changing properties of graphs 101

Computer methods for CAS Activities in Books A–D **103**

Solutions to Computer activities **107**

Maxima reference guide **142**

Acknowledgements **151**

Index **153**

Introduction

During this module you will use the Maxima computer algebra system (CAS) introduced in MST124 to investigate mathematical concepts and perform calculations that may be too laborious or difficult to do by hand.

This *Guide* is designed to be read in conjunction with the main study texts and is not meant to be read all at once because its later sections use mathematical concepts that you may not have yet met.

As you work through the main study texts in Books A–D, you will come across sections or activities marked with the icon shown here. At that point you will be directed to attempt an activity using Maxima or to study a section of this *Guide*. The mathematical solutions to activities in Books A–D that require the use of Maxima are given in those books, but details of the computer methods needed are given in the section titled 'Computer methods for CAS activities in Books A–D', towards the end of this *Guide*. The section following that, titled 'Solutions to Computer activities', contains Maxima statements and comments for the activities in this *Guide*.

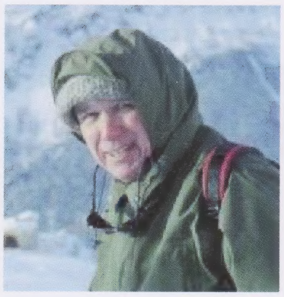
Section 1 gives an overview of Maxima and includes activities in which you can install, test and configure Maxima on your computer. If you already have a working copy of Maxima ready to use, you can skip this section.

Section 2 describes how to use the Maxima user interface for basic calculations and Section 3 gives an overview of using Maxima for manipulating algebraic expressions, plotting graphs and calculus. These sections relate to Unit 1 of MST125 and revise skills introduced in MST124. If it is some time since you studied MST124, or you have never studied it, you may want to work through these sections in detail; otherwise you should use them to refresh your memory as necessary. If you require further details on any of the topics covered, you might like to read the appropriate section of the MST124 *Computer algebra guide* available on the OU Maxima website at `learn1.open.ac.uk/site/maxima`. If you are studying MST124 and MST125 starting at the same time, you won't need to study these sections, as MST124 will introduce the concepts as you need them.

The 'Maxima accessibility guide' (Section 10) is primarily aimed at students who may have difficulty using Maxima because of a disability, and provides some advice and suggestions.

The final section of this *Guide* contains a summary of all the Maxima commands introduced. This should be useful for reference, both as you work through the module and afterwards.

Note that this *Guide* does not attempt to describe all the features of Maxima. If you need more information, please consult the extensive documentation provided on the Maxima website at `maxima.sourceforge.net` or use Maxima's help system, which is described in Subsection 2.10.



William Schelter (1947–2001)

Maxima's origins lie in a system called Macsyma, a well-respected computer algebra system first developed in the late 1960s at the Massachusetts Institute of Technology and funded by the United States Department of Energy. From 1982 until his sudden death in 2001, William (Bill) Schelter, a professor of mathematics at The University of Texas at Austin, maintained a version of Macsyma which we now know as Maxima. In 1998 he obtained permission from the Department of Energy for the system to be made freely available to all. Maxima is now dedicated to his memory.

A word of warning

Using a computer algebra system is not a substitute for learning the mathematical techniques taught in this module! You need to become proficient at using the methods taught so that you can apply them in the examination and in your future studies, and also to help you develop your mathematical understanding and intuition.

Maxima is not intelligent – it is just a tool, similar to your calculator. You need to use your own mathematical skills to guide it and interpret its results critically.

Conventions

Here are the various computing terms used in this *Guide*, and how they should be interpreted on Microsoft Windows and Apple Mac computers. The *Guide* concentrates on using Maxima on a Microsoft Windows computer, although Maxima is available for many different types of computer.

Computer terminology

Term	Microsoft Windows	Apple Mac
Click	Click the left mouse/touchpad button	Click the mouse/touchpad button
Right click	Click the right mouse/touchpad button	Hold the Ctrl key while clicking the mouse/touchpad button
Select	*Either* click to select an option, *or* select text by dragging the cursor over the text while pressing the left mouse button	*Either* click to select an option, *or* select text by dragging the cursor over the text while pressing the mouse button
Ctrl-Q Etc.	Hold down Ctrl while pressing Q Etc.	Hold down Command while pressing Q Etc. – but there are exceptions to this pattern that we'll mention as they arise

On some computer keyboards the Enter key might be labelled Return, or simply ↵, and the Shift key might be labelled ⇧.

In this *Guide*, these keys are referred to as Enter and Shift, respectively.

Sometimes Enter and Return are two separate keys. For such keyboards references in this *Guide* to the Enter key should be interpreted as references to the Return key.

1 Installing, testing and configuring Maxima

If you already have a working version of Maxima installed on your computer, for example from your study of a previous module, then you can continue to use that version and can skip this section.

1.1 Introducing and installing Maxima

Maxima can be used on many different types of computer, and there are several different intermediary systems, known as **interfaces** or **front-ends**, that provide a means of communication between you, the user, and the Maxima system itself. This is illustrated in Figure 1. The interfaces enable you to enter commands that are then processed by Maxima, and they display the results obtained.

All the commands mentioned in this *Guide* can be used with any interface to Maxima, unless indicated otherwise.

The recommended interface is **wxMaxima**, which is shown in Figure 2. (The letters 'wx' in the name of this interface refer to the software used to create it.)

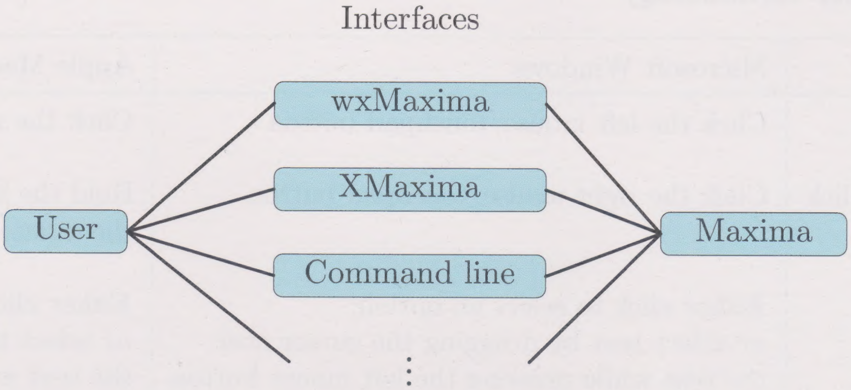
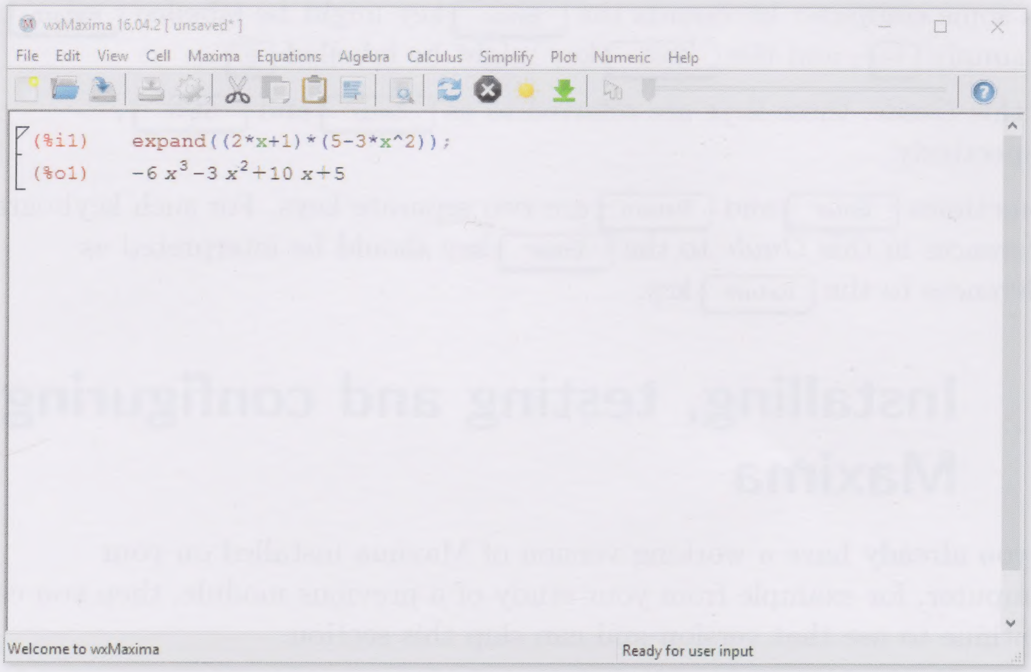


Figure 1 Different interfaces for Maxima


wxMaxima 16.04.2 [unsaved*]
File Edit View Cell Maxima Equations Algebra Calculus Simplify Plot Numeric Help

```
(%i1)  expand((2*x+1)*(5-3*x^2));
(%o1)  -6 x^3 -3 x^2 +10 x+5
```

Welcome to wxMaxima

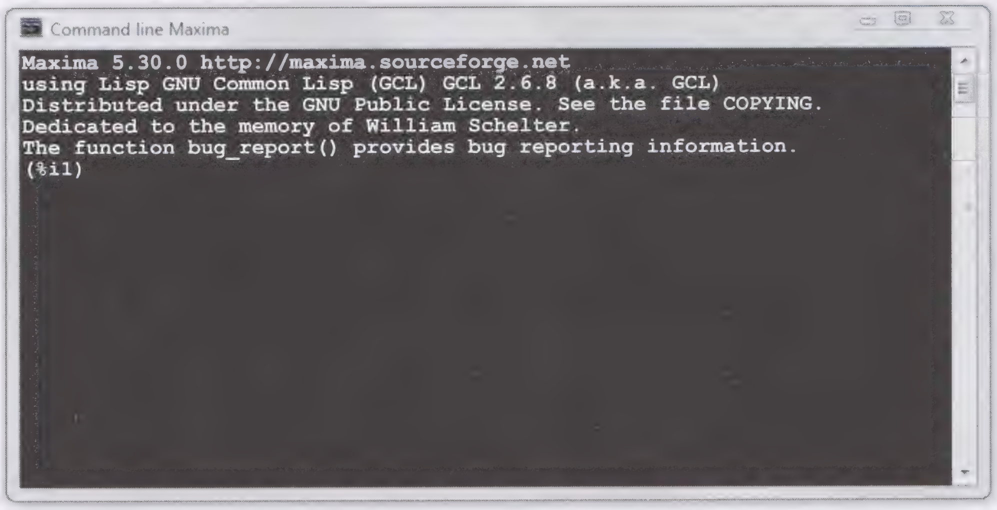
Ready for user input

Figure 2 The wxMaxima interface

If you need to use an interface other than wxMaxima, for example if you are not using a Windows computer and you are unable to install wxMaxima, then you should try to achieve all the outcomes shown in this *Guide* using your chosen interface.

If you use a screenreader, then the basic **command-line** interface shown in Figure 3 may be the easiest interface for you to use.

If you think you may have difficulty using the wxMaxima interface, because of a disability for example, then you should read Subsection 10.1 now. This contains details of how wxMaxima might be configured to meet your needs, and should help you decide whether to use wxMaxima or the command-line interface.



```
Maxima 5.30.0 http://maxima.sourceforge.net
using Lisp GNU Common Lisp (GCL) GCL 2.6.8 (a.k.a. GCL)
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
The function bug_report() provides bug reporting information.
(%i1)
```

Figure 3 The command-line interface to Maxima

Before you can use Maxima, you need to install it on your computer. You will do this in the next activity.

Computer activity 1 Installing Maxima

(a) Go to the OU Maxima website at `learn1.open.ac.uk/site/maxima` and follow the instructions there to download and install Maxima on your computer. (You can also find a link to the OU Maxima website on the MST125 website.)

(b) Create a folder on your computer in which to store your Maxima work for this module. You might like to put this on your computer desktop, or in your usual documents area.

If you are studying within a prison or other closed institution, you can install Maxima from the *Offline resources* disc.

1.2 Starting, testing and configuring wxMaxima

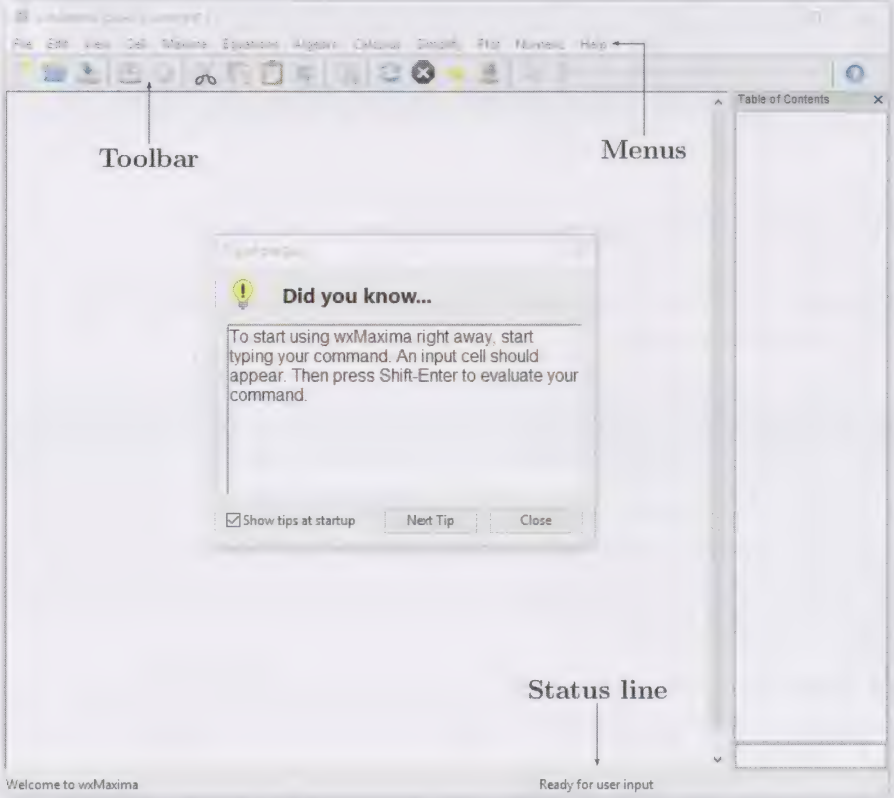
If you are using the command-line interface you should study Subsection 10.2 instead of this subsection. If you are using a different interface, you should make sure that you can achieve similar outcomes with that interface.

Computer activity 2 Starting wxMaxima

Start wxMaxima on your computer, and keep it open to work with as you study this section.

If you cannot remember how to do this on your computer, see the OU Maxima website: `learn1.open.ac.uk/site/maxima`.

The wxMaxima interface to Maxima, illustrated in Figure 4, consists mainly of a large white working area (called the **worksheet**) where you can type Maxima commands. These commands instruct Maxima to either perform calculations or change system settings.



Did you know...

To start using wxMaxima right away, start typing your command. An input cell should appear. Then press Shift-Enter to evaluate your command.

☑ Show tips at startup

Welcome to wxMaxima

Ready for user input

Figure 4 The initial wxMaxima window

When the software first starts, a secondary window showing the 'Tip of the Day' is also displayed. You can close it by clicking the 'Close' button. If you want to prevent this window from appearing the next time you start wxMaxima, click the box in its bottom left-hand corner, so that it is no longer ticked, before clicking 'Close'. (You can always reinstate these messages at any time by selecting `Show tips` from the `Help` menu.)

Along the top of the main window (or along the top of the screen in the case of an Apple Mac) is a list of menu names, including the `File` menu which allows you to save or print your work and re-open previously saved sessions (there is more on this later). You can also close the software from the `File` menu by selecting `Exit`. (On an Apple Mac you can do this by selecting `Quit wxMaxima` from the `wxMaxima` menu.)

In Figure 4, a row of icons known as the **toolbar** is shown below the row of menu names. This is displayed only if `Main Toolbar` is selected in the `View` menu. The icons provide shortcuts to commonly used functions.

The `View` menu also allows you to display palettes of commonly used commands. You can use these palettes as an alternative to typing some of the commands given in this *Guide*.

Also shown in Figure 4 is the **Table of Contents**. This shows the names of any sections or subsections that you may add to your worksheet (see Subsection 2.8). You may want to close the Table of Contents, by clicking the cross near its top right-hand corner.

At the bottom right-hand corner of the wxMaxima window is a message that indicates the current status of the Maxima system. When Maxima is waiting for instructions from you this reads 'Ready for user input'. Other possible messages include 'Maxima is calculating', or 'Reading Maxima output'.

The following activity shows you how to use wxMaxima to perform a simple calculation and checks that your system is working properly. Follow the steps using wxMaxima yourself as you read the activity.

If you encounter unexpected problems when working through this activity, check the Frequently asked questions (FAQs) section on the OU Maxima website: `learn1.open.ac.uk/site/maxima`.

Computer activity 3 Your first Maxima calculation

Follow these steps to use Maxima to calculate $2 + 3 \times \frac{4}{5}$.

(a) Click anywhere on the large white area of the wxMaxima window, to ensure that the text you type next is directed to the correct place.

(b) Type the following exactly as it appears here:

```
2+3*4/5;
```

including the semicolon (`;`) at the end.

Notice that as you type your command, it is shown in wxMaxima preceded by a red arrow: `-->`.

This is the **input prompt** and shows where commands can be typed.

To the left of the input prompt is the **cell marker**

which will be described in more detail later.

As you type, the colour of the text changes to highlight the different elements of your expression: numbers are coloured differently to the operations.

(c) Press **`Shift-Enter`**, that is, hold down the Shift key while pressing Enter .

(Remember: if, as on some Macs, your keyboard has separate Enter and Return keys, then you should hold down the Shift key while pressing Return .)

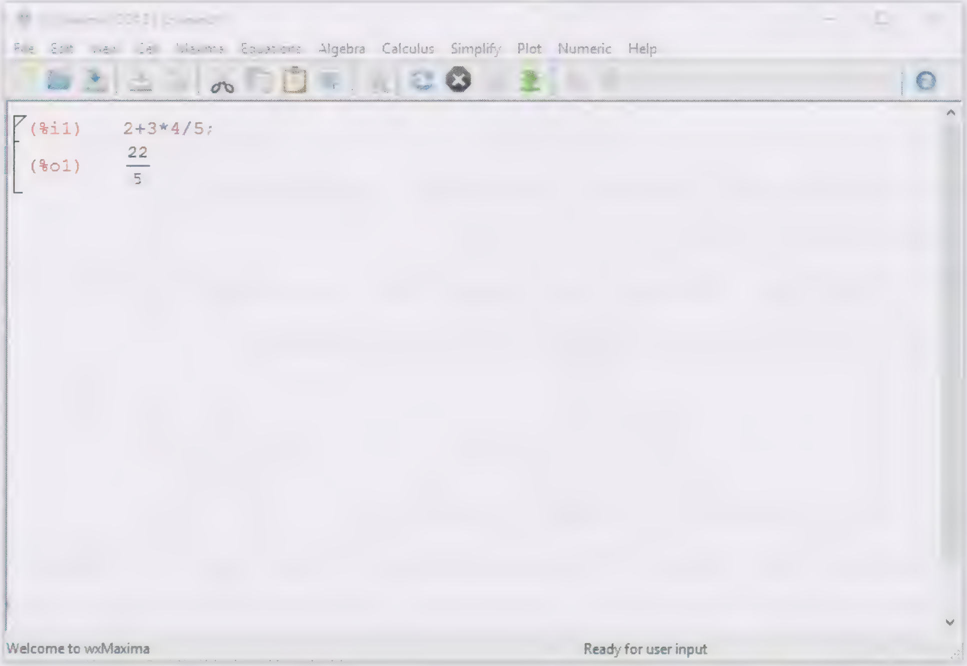
If you make a mistake, try clicking elsewhere on the worksheet, below your error, and try again.

Pressing **`Shift-Enter`** instructs wxMaxima to send the expression to Maxima to calculate. This is known as **evaluating** the expression.

The input prompt (`-->`) is now replaced by `(%i1)`. This is how Maxima labels lines. The `i` stands for 'input': this is input line number 1. Within each Maxima session line numbers start at 1 and increase for each new input line. The percentage symbol (`%`) is used by Maxima to indicate objects built into the system. There is more about this later.

The status message in the bottom right-hand corner of the wxMaxima window first changes to 'Maxima is calculating', then 'Reading Maxima output', before the result of the calculation is displayed. The message then displays 'Ready for user input' when the system is ready to accept another command.

The result is displayed under the input line, preceded by the label `(%o1)`. The `o` stands for 'output', so this label identifies the line as output line 1. It gives the output corresponding to input line 1.



```
(%i1)    2+3*4/5;
          22
(%o1)     --
          5
```

Welcome to wxMaxima

Ready for user input

If, in Computer activity 3, Maxima behaved as described, then you have a working system to use throughout MST125. Before finishing though, you should make a couple of small changes to how the wxMaxima interface behaves.

In Computer activity 3 you used `Shift-Enter` rather than Enter to instruct Maxima to evaluate expressions, and the input prompt (`-->`) appeared only once you started typing. Both these behaviours can be disconcerting, but can be changed, as described in Computer activity 4.

It is recommended that you configure wxMaxima as described in Computer activity 4, and in the remainder of this *Guide* it is assumed that you have done so.

Computer activity 4 Configuring wxMaxima

(a) From the `Edit` menu, select `Configure` (or on an Apple Mac computer, from the `wxMaxima` menu, select `Preferences`).

Alternatively, click the following icon on the toolbar.

This opens the ‘wxMaxima configuration’ window.

(b) Within the Configuration window, click the icon labelled ‘Worksheet’ to access the worksheet settings, then tick the ‘Enter evaluates cells’ box by clicking it.

This changes the behaviour so that pressing Enter alone evaluates commands.

(c) In the same list of settings, tick the ‘Open a cell when Maxima expects input’ box by clicking it.

This ensures that the input prompt is shown whenever Maxima is waiting for input.

(d) In the same list of settings, ensure the ‘Show user-defined labels instead of (%oxx)’ box is *not* ticked.

This ensures that the output line numbers are always shown.

(e) Click on the icon labelled 'Options' in the Configuration window, then tick the 'Save sidebar status' box by clicking it.

This ensures that your preferred configuration of the wxMaxima window, for example hiding the Table of Contents, is retained between Maxima sessions.

(f) Click 'OK' (or close the window if using an Apple Mac).

Closing wxMaxima

When you have finished using Maxima, you can close Maxima and the wxMaxima interface by using one of the following methods.

- Select `Exit` from the `File` menu (or `Quit wxMaxima` from the `wxMaxima` menu on an Apple Mac computer).
- Press `Ctrl-Q`. (Here, `Q` stands for quit.)
- On a Windows computer, click the usual small cross button at the top right-hand corner of the wxMaxima window.

You may be asked if you want to save your changes to the worksheet before closing. At this stage, click 'No'. You will see how to save your work in Subsection 2.9.

2 Getting started with Maxima

In this section you will revise how to use the wxMaxima interface to perform calculations with Maxima. If you are familiar with Maxima from MST124, you should be able to quickly skim through this section, and Section 3, and then try some of the activities in Section 3 to check your skills, revising them more thoroughly as necessary.

If you require further details on any of the topics covered, you might like to read the appropriate section of the MST124 *Computer algebra guide* available on the OU Maxima website: `learn1.open.ac.uk/site/maxima`.

If you encounter unexpected problems when working through this section, remember to check the Frequently asked questions (FAQs) section on the OU Maxima website.

2.1 Calculating with Maxima

The way in which a calculation or a command has to be entered in Maxima is known as its **syntax**. The Maxima syntax for the basic mathematical operations and other functions is summarised below. You can practise using some of these commands in Computer activity 5.

In this *Guide* we will use ^ for powers. Many Maxima commands take the form of a command name, such as `sqrt` or `abs`, followed by a pair of round brackets containing one or more objects, such as numbers, that the command operates on. Such an object is called an **argument** of the command. In the summary tables in this *Guide*, such as the one below, arguments of commands are often represented by ▮.

Like many computer systems, Maxima assumes that all angles are measured in radians. The names of the inverse trigonometric functions $\sin^{-1}$, $\cos^{-1}$ and $\tan^{-1}$ in Maxima are `asin`, `acos` and `atan` respectively. This notation is used by many computer systems. It is short for arcsin, arccos and arctan, the alternative names often used for $\sin^{-1}$, $\cos^{-1}$ and $\tan^{-1}$.

Mathematical operations and functions

Operation or function	Syntax	Example
Addition	`+`	`4+3;`
Subtraction	`-`	`4-3;`
Multiplication	`*`	`4*3;`
Division	`/`	`4/3;`
Brackets	`(` *and* `)`	`2*(3+4);`

Powers, for example 2^3	`^`
	or `**`
		`2^3;`
		`2**3;`

Square root, for example $\sqrt{5}$	`sqrt(▮)`	`sqrt(5);`

Exponential, for example e^3	`%e^▮`
	or `exp(▮)`
		`%e^3;`
		`exp(3);`

Natural logarithm, ln	`log(▮)`	`log(8);`
Magnitude (absolute value)	`abs(▮)`	`abs(-3);`
sin	`sin(▮)`	`sin(1);`
cos	`cos(▮)`	`cos(3*%pi/2);`
tan	`tan(▮)`	`tan(%pi/4);`
cosec	`csc(▮)`	`csc(2);`
sec	`sec(▮)`	`sec(%pi/3);`
cot	`cot(▮)`	`cot(%pi/4);`
$\sin^{-1}$	`asin(▮)`	`asin(sqrt(3)/2);`
$\cos^{-1}$	`acos(▮)`	`acos(1/sqrt(2));`
$\tan^{-1}$	`atan(▮)`	`atan(1/2);`

The symbols `%e` and `%pi` in the table above are the Maxima representations of the mathematical constants e and π. The `%` symbol indicates that this is the name of a special quantity built into Maxima. When displaying mathematics in an output line, wxMaxima displays `%pi` as π.

Mathematical constants

Constant	Syntax	Example
e	`%e`	`%e^2;`
π	`%pi`	`2*%pi;`

There is no Maxima command for logarithms to bases other than e. Logarithms to other bases can be found using:

$$\log_b x = \frac{\log_a x}{\log_a b}.$$

In particular, $\log_{10} x$ can be calculated in Maxima using

```
log(x)/log(10)
```

You should end all Maxima commands with either a semicolon (`;`) or a dollar sign (`$`). Ending a command with a dollar sign tells Maxima to perform the instruction, but not to display the answer. When using wxMaxima, if you forget to finish your command with a semicolon or dollar sign, then the system will add a semicolon for you. Other interfaces may not do this. In such systems, pressing Enter without ending the line with a semicolon or dollar sign simply moves the editing cursor to the next line, waiting for you to finish entering a complete command.

You can also enter several commands on one line, ending each one with a semicolon or dollar sign.

Maxima generally ignores all spaces that you type while entering a calculation or command, so you can use spaces to make a command easier to read. Once wxMaxima has been configured to use Enter to evaluate commands (as described in Computer activity 4), you can then use `Shift-Enter` to enter line breaks, which are also ignored by Maxima.

Warning: Do not omit multiplication signs!

A common error when first using Maxima is to omit multiplication signs. For example, you might type

- `2sqrt(2)` rather than `2*sqrt(2)`, or
- `2(3+4)` rather than `2*(3+4)`.

Doing this will result in an error. The displayed error message might include one of the following statements:

- *parser: incomplete number; missing exponent?*
- *incorrect syntax: Syntax error*
  incorrect syntax: Too many)'s
  incorrect syntax: Premature termination of input at ;.
- *... is not an infix operator.*

If you receive one of these error messages, first check for missing multiplication signs!

Warning: Take care with question marks (?)

Do not type question marks when asking Maxima to evaluate an expression. The symbol `?` is a special symbol in Maxima which provides access to underlying systems.

The one exception to this is when you are using the commands for obtaining help from the system, which you will see in Subsection 2.10.

You can practise using Maxima for mathematical calculations in the following activity.

Computer activity 5 Calculating with Maxima

Calculate the following using Maxima.

(a) $\sqrt[5]{32}$

Since $\sqrt[5]{32} = 32^{\frac{1}{5}}$ this can be found by entering

```
32^(1/5);
```

(The brackets in this expression are required since Maxima calculates expressions using the correct order of mathematical precedence.)

Remember to press Enter to evaluate the command.

(b) $\sin\left(\frac{\pi}{3}\right)$

Remember that to enter the constant π in Maxima, you type `%pi`.

Maxima returns an exact answer, in the form of a surd.

(c) $\tan 30°$

Remember to convert the angle to radians, by multiplying 30 by $\frac{\pi}{180}$.

(d) e^4

Remember, the Maxima symbol for e is `%e`, or you can use the `exp(□)` function.

Since there is no simpler exact way of expressing e^4, Maxima leaves this expression unevaluated.

(e) ln 3.4

Remember that the Maxima command for a natural logarithm is `log(▒)`. Maxima has no command `ln(▒)`.

Here, since the argument is a decimal number, Maxima returns a decimal approximation to the answer.

During the previous activity you may have noticed that when you enter an opening bracket, the wxMaxima interface automatically adds a closing one for you. This behaviour can be turned off by unticking ‘Match parenthesis in text controls’ in the ‘Worksheet’ settings of the Configuration window. (See Computer activity 4 for how to open the Configuration window.)

2.2 wxMaxima cell markers

Corresponding matching input and output lines in wxMaxima form a **cell** identified with the marker shown in Figure 5.

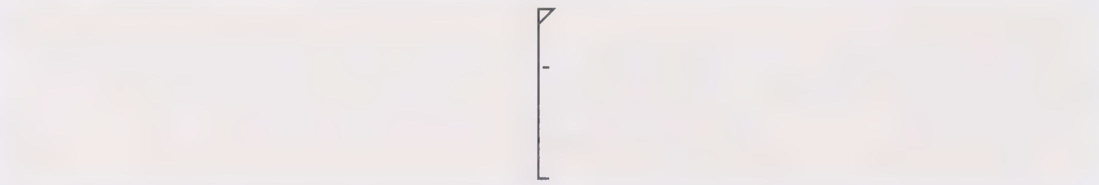


Figure 5 The wxMaxima cell marker

The top part of the cell marker (between the upper two horizontal marks) corresponds to the input line(s), and the bottom part corresponds to the output line(s). The cell marker turns red while an input line is being edited, and is shown surrounded by a box while the cell is being evaluated, as illustrated in Figure 6.

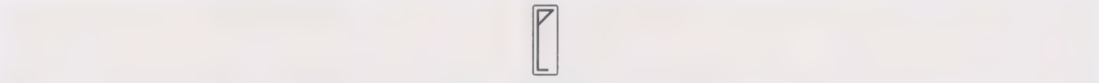


Figure 6 The cell marker during evaluation

Cells can be compressed (and uncompressed) to help make a complicated worksheet easier to read by clicking on the small triangle at the top of the cell marker. A cell can be deleted from a worksheet by clicking part of the cell marker other than the top triangle to highlight it, then pressing the Delete keyboard key.

2.3 Decimal numbers

In Computer activity 5 you saw that Maxima usually tries to calculate quantities *exactly* rather than approximating them by decimal numbers. It does this by manipulating the symbols in the expression according to the rules of mathematics. This is sometimes known as **symbolic computation**. If an exact answer cannot be obtained, the expression will be returned unevaluated, as in Computer activity 5(d). The exception to this is if the expression to calculate contains a decimal number, such as in Computer activity 5(e), when Maxima will work in terms of decimal numbers.

If you wish to have a result displayed as a decimal number, you can instruct Maxima to do so with the `float` command. The name of the command `float` arises from the fact that the method used by computers to store decimal numbers internally using the binary digits 0 and 1 is called a *floating-point representation.* The `float` command instructs Maxima to convert a number stored symbolically as a mathematical expression to one stored using a floating-point representation.

Converting to decimal numbers

Operation	Command	Example
Convert to a decimal number	`float(□)`	`float(sqrt(2));`

Maxima usually displays decimal numbers to 16 significant figures (though it suppresses trailing zeros at the end of the decimal part of a number; for example 0.125 is displayed simply as `0.125` rather than as `0.1250000000000000`). This is also the precision to which Maxima performs decimal calculations. You will see in Subsection 2.7 how to change the number of significant figures displayed.

You may also see decimal numbers displayed in the form

```
6.223015277861141 10^139
```

which is wxMaxima's way of displaying scientific notation. It means $6.223\,015\,277\,861\,141 \times 10^{139}$. (The command-line interface would display this as `6.223015277861141E+139`.) You can also enter numbers given in scientific notation. For example, to enter 6.022×10^{23}, you can type

```
6.022e23
```

(Note that the use of `e` here, without multiplication or percentage symbols, is different to the use of `%e`, which represents the mathematical constant e.)

Computer activity 6 Simplifying and approximating numerical expressions

For each of the following expressions, use Maxima to simplify it and then find a decimal approximation for it.

(a) $\sqrt{325}$ (b) 5^{200}

The expression in part (b) doesn't involve decimal numbers and so the result is evaluated exactly. Notice, however, that the middle 80 digits of the answer are not displayed because the system does not expect you to read them all!

2.4 Troubleshooting wxMaxima

If you are using wxMaxima and it does not seem to be responding, try the following suggestions.

(Suggestions for troubleshooting Maxima when using the command-line interface are given in Subsection 10.3.)

Troubleshooting wxMaxima

Cell Marker	Status message	Comments
Boxed: ▯	Maxima is calculating	Maxima is evaluating a command that is taking a long time to complete.
Boxed: ▯	Ready for user input	Maxima is waiting for you to type something before continuing. In either case you might wish to abort the command in one of the following ways. • Click ⊗ on the toolbar. • Select `Interrupt` from the `Maxima` menu. • Type `Ctrl-G` (on an Apple Mac hold down the [Command] key while pressing [.]). Maxima might take some time to respond to this request.
	Maxima process terminated	Maxima (but not the wxMaxima interface) has stopped working. Select `Restart Maxima` from the `Maxima` menu.

If you encounter other problems, check the Frequently asked questions (FAQs) section on the OU Maxima website: `learn1.open.ac.uk/site/maxima`.

2.5 Reusing and editing commands

In Computer activity 6 you may have entered each expression twice: once to evaluate it exactly and once for a decimal answer. This is, however, not necessary. When you have entered and evaluated an expression, you can edit it and then evaluate it again, as shown in the following activity.

(Details on reusing and editing commands when using the command-line interface are given in Subsection 10.4.)

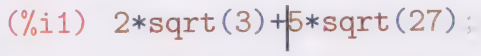
Computer activity 7 Editing a command

(a) Calculate $2\sqrt{3}+5\sqrt{27}$ by entering the following line into Maxima.

```
2*sqrt(3)+5*sqrt(27);
```

(b) Edit your entry in part (a) to calculate $2\sqrt{3}-5\sqrt{27}$ as follows.

(i) Click on the expression you entered in part (a), or move to it using the up and down keyboard arrow keys.

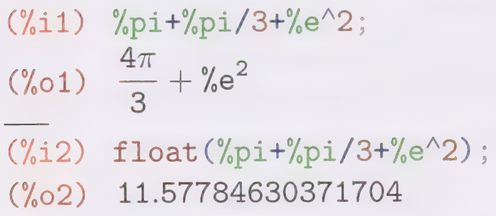
> As you do this the cell marker will turn red and you will see a small flashing vertical line (the **editing cursor**) appear in the expression.
>
> 
>
> ```
> (%i1) 2*sqrt(3)+|5*sqrt(27);
> ```

(ii) Edit the expression by using the ← (backspace) or Delete key to delete the + sign and enter a - sign instead.

(iii) Press Enter .

> Pressing Enter re-evaluates the cell. When a cell is re-evaluated in this way, the input and output line numbers are updated as if a new cell had been added.

Should you click in the space *between* two cells, a horizontal cursor will be shown, as in Figure 7. This indicates the position at which a new cell will be created, if you begin typing here. This is also shown as you move between cells using the up and down keyboard arrow keys.



(%i1) %pi+%pi/3+%e^2;

(%o1) $\frac{4\pi}{3}+\%e^{2}$

(%i2) float(%pi+%pi/3+%e^2);

(%o2) 11.57784630371704

Figure 7 The wxMaxima horizontal cursor, positioned between two cells

In wxMaxima you can also copy and paste expressions, or parts of expressions, as you might copy and paste when using other software; that is, using `Copy` and `Paste` from the `Edit` menu, or the `Ctrl-C` and `Ctrl-V` keyboard shortcuts.

Maxima also permits the result of one calculation to be used in another calculation. The symbol `%` represents the result of the last command evaluated. Due to the updating of line numbers when a command is re-evaluated, this might not be the last command in the worksheet, but it will be the command with the highest line number. So, for example, if you enter `sqrt(%);` Maxima calculates the square root of the last evaluated output. You can also use the output (or input) of other lines, by typing the line label in a calculation. For example, if you enter `%o5^3;` Maxima calculates the cube of the expression in output line 5.

Computer activity 8 Using and editing previous calculations with wxMaxima

(a) Enter `cos(5*%pi/6);` and press Enter to evaluate it.

(b) On the following line type `2*%;` and press Enter.

Maxima returns $-\sqrt{3}$, which is twice the previous answer.

(c) Edit the line where you typed `cos(5*%pi/6);`, change it to `sin(5*%pi/6);` and re-evaluate the cell by pressing Enter.

Notice that the line number and output of the line you edited is updated, but the result of `2*%;` does not change. This is because when the `2*%;` cell was evaluated, the symbol `%` represented the result of the old version of the edited command.

(d) From the `Cell` menu, select the `Evaluate All Visible Cells` option, or press `Ctrl-R`.

This re-evaluates all visible cells in the worksheet, *in the order in which they appear*, updating all the line numbers as it does so.

Notice that the result of `2*%;` has now changed. At the time it was re-evaluated the symbol `%` represented the value of the last evaluation, which was the revised version of the previous line.

(e) In a new cell at the bottom of your worksheet enter `%o░^2;` with ░ replaced by the output line number corresponding to the result of `sin(5*%pi/6);` entered earlier.

This command finds the square of the result of the line referenced.

Re-evaluating a wxMaxima worksheet

Selecting the `Evaluate All Visible Cells` option from the `Cell` menu re-evaluates all visible cells in the worksheet, in the order in which they appear.

This can also be achieved by pressing `Ctrl-R`.

By default, the re-evaluation of a worksheet will stop if an error is encountered. This behaviour can be changed so that evaluation continues past the error by unticking ‘Abort evaluation on error’ in the ‘Maxima’ settings of the Configuration window.

2.6 Variables

Maxima allows you to assign a value to a *variable*, and then use that variable in further calculations.

For example, you can assign the value 23 to the variable `a`, and then use `a` in subsequent expressions or commands, such as $\mathtt{a}^2 - 3\mathtt{a} + 2$. You can also assign expressions, both numerical and (as you will see later) algebraic, and indeed any other Maxima object, to a variable, and use that variable in subsequent commands.

To assign a value or expression to a variable you use a colon (`:`). For example, the command

```
a:23;
```

assigns the value 23 to the variable `a`. The colon can be read as ‘is assigned the value’.

Warning

You cannot use `=` to assign a value to a variable. This symbol is used to define *equations* in Maxima.

Variable names can be any combination of letters and numbers that begins with a letter. For example, Maxima will accept any of the following variable names.

```
a    A    solution    solution2    x2b
```

Some words are built-in to Maxima and cannot be used as variable names. If you try to assign such a variable, you will get an error such as:

```
assignment:  cannot assign to ...
-- an error.  To debug this try:  debugmode(true);
```

If you encounter such an error, choose a different name for your variable.

Variable names are *case-sensitive*; for example, the variable `x` is different from the variable `X`. If a variable name is the name of a Greek letter, then wxMaxima will display it in output lines as the proper Greek character; for example, the variable `alpha` will be displayed as α.

Once a variable has been assigned a value, Maxima will remember this value for the rest of your session. This may sometimes surprise you. It is easy to forget that earlier in the day you assigned a value to, say, `x`; you may then be confused later on when a calculation involving `x` gives an unexpected result. You can, however, tell Maxima to 'forget' about a variable to which you have previously assigned a value. You can also ask it to list all the variables that you have assigned values to. These commands are demonstrated in the following activity.

Computer activity 9 Working with variables

(a) Assign the value of $\sqrt{8}$ to the variable `a` by entering

```
a:sqrt(8);
```

The value of `a` is displayed as output, in an equivalent form.

Note that when assigning variables, wxMaxima precedes the output with either the output line number or the name of the variable being assigned a value. Which is shown is determined by whether or not 'Show user-defined labels instead of (%oxx)' is ticked in the 'Worksheet' settings of the Configuration window. In this *Guide*, the output line numbers are shown.

(b) Enter `a;`

This displays the current value of `a`.

(c) Assign the value of `a`$\sqrt{2}$ to `b`.

Don't forget to include a multiplication sign when inputting `a`$\sqrt{2}$.

The value of `b` is simplified when displayed.

(d) Edit the line where you assigned $\sqrt{8}$ to `a`, so that the value $\sqrt{7}$ becomes assigned to `a`.

(e) Enter `b;` to display the value of `b`.

Notice that the value of `b` has not changed. The variable `b` was defined when `a` had its original value.

(f) Enter `values;` to list the names of all the variables that are currently assigned values.

The variable names are given within square brackets, separated by commas. This is how Maxima displays *lists*.

(g) Enter `kill(a);` to remove the variable `a` from the system.

You should obtain the output *done*.

(h) Enter `a;` to display the value of `a`.

The variable name is output, as `a` no longer has a value.

(i) Enter `b;` to display the value of `b`.

`b` is still defined.

Notice that in Computer activity 9(d) and (e), when the value of `a` was changed, the value of `b` was not affected, since it was defined when `a` had its original value. If you want to update the value of `b` using a different value of `a`, then you should enter a new value of `a` then re-evaluate the line defining `b`. You can do this in wxMaxima by selecting the line and pressing Enter, or re-evaluating the whole worksheet by selecting the option `Evaluate All Visible Cells` from the `Cell` menu.

Working with variables

Operation	Command	Example
Assign a value to a variable	`variable : value`	`a:23;`
Display the value of a variable	`variable`	`a;`
List all user-assigned variables	`values`	`values;`
Remove an assigned variable	`kill( variable )`	`kill(a);`
Remove all assigned variables	`kill(all)`	`kill(all);`

Note: here the placeholder `variable` represents any variable name.

You can practise using variables in the following activity.

Computer activity 10 Using variables

(a) Define the variable `a` to have the value 42.

(b) Define the variable `b` to be equal to $\sin^2 \texttt{a} + \cos^2 \texttt{a}$.

To enter, for example, $\sin^2 \texttt{a}$ in Maxima, you have to type `sin(a)^2`

(c) Find the decimal approximation of the value of `b`.

2.7 System variables

There are some variables built into Maxima whose values affect the behaviour of the system. These are called **system variables**. You met one system variable in Computer activity 9: the variable **`values`** holds a list of the names of all the variables you have defined.

Another system variable, **`fpprintprec`**, specifies the number of significant figures of decimal numbers that are displayed. The name `fpprintprec` is an abbreviation of ‘floating-point print precision’.

To change the system behaviour so that, for example, only 4 significant figures are displayed, use the command `fpprintprec:4;` to assign the value 4 to the variable `fpprintprec`.

You can set the variable `fpprintprec` to any value between 2 and 16. Also, setting it to 0 restores the default behaviour of displaying 16 significant figures. The value of `fpprintprec` sets the *requested* number of significant figures, but Maxima does not always exactly meet the request.

Note that setting `fpprintprec` only affects how decimal numbers are displayed: it does not change the accuracy with which calculations are performed.

Computer activity 11 Displaying a specified number of significant figures

Display the decimal approximation of π to 8 significant figures by doing the following.

(a) Set the system variable `fpprintprec` to be 8.

(b) Display the decimal approximation of π.

You can reset the values of `fpprintprec` and all other system variables to their original values by using the `reset()` command, which has no arguments. Try entering `reset();` followed by `float(%pi);` and check that the value of π is displayed to 16 significant figures. Note that the `reset` command also resets your Maxima line numbers to start at 1 again.

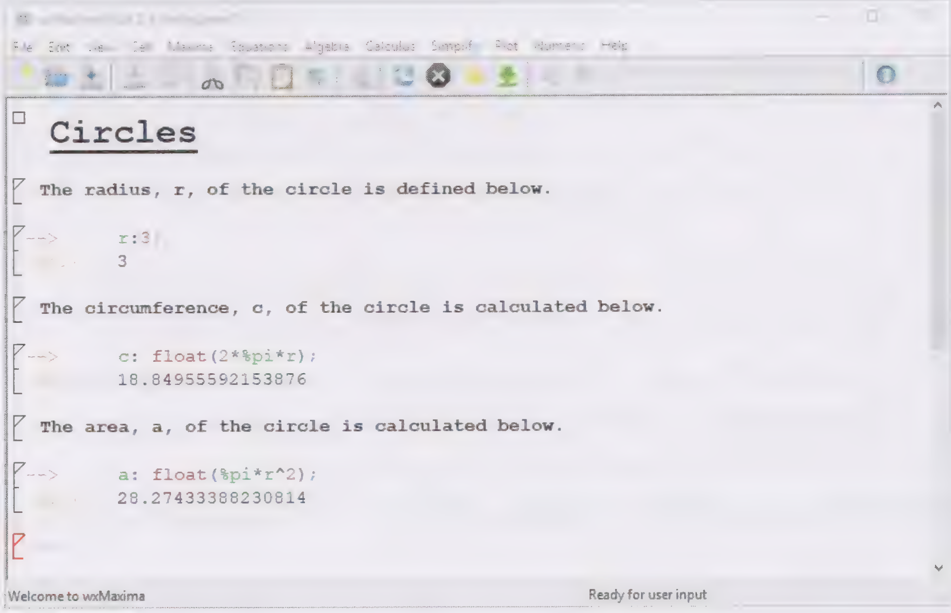
Resetting system variables

Operation	Command	Example
Reset all system variables	`reset()`	`reset();`

Note that the list of variables displayed when you enter `values;` does not include any system variables. If one of your variables does not appear in the list displayed by `values;`, then you have probably used a variable name that is also the name of a system variable and hence changed the value of that system variable. In extreme circumstances this may change the behaviour of Maxima, which can be restored by resetting all system variables as described above.

2.8 Annotating your wxMaxima worksheet

It is important to be able to clearly explain and present your mathematical work, both so that someone else can understand it, and so you can understand what you have done if you return to your work after some time. To help you do this, wxMaxima allows you to enter text comments within your worksheet, as illustrated in Figure 8.



Circles

The radius, r, of the circle is defined below.

```
-->     r:3
        3
```

The circumference, c, of the circle is calculated below.

```
-->     c: float(2*%pi*r);
        18.84955592153876
```

The area, a, of the circle is calculated below.

```
-->     a: float(%pi*r^2);
        28.27433388230814
```

Figure 8 Adding text to a wxMaxima worksheet

To enter text into your worksheet, first you need to create a suitable cell in which to type. To do this, position the cursor and then select the appropriate one of the following commands from the `Cell` menu.

- `Insert Text cell`

- `Insert Title cell`
- `Insert Section cell`
- `Insert Subsection cell`

The new cell will be created under the cell in which the editing cursor appears, or between two existing cells at the position indicated by the horizontal cursor, if shown. Remember, the horizontal cursor appears whenever you click between two cells, or move within the worksheet using the up and down keyboard arrow keys.

As you type your text, to start a new line within the cell press Enter. When you have finished typing your text, click elsewhere on the worksheet or use the up or down keyboard arrow keys to move out of the text cell. If your text cell is at the bottom of the worksheet, and you click or move below it, an input prompt may not be shown until you start typing again.

Section and Subsection cells are automatically numbered.

Each Title cell, Section cell and Subsection cell has a small square in its top left corner. Clicking this square hides (or reveals) the contents of the worksheet from that point onward, until the next cell of the same type.

Whenever you add a Section or Subsection cell, a corresponding entry is added to the Table of Contents window. You can display this window by selecting `Table of Contents` from the `View` menu, or by pressing `Alt-Shift-T`, that is, by holding down the Alt and Shift keys while pressing T.

Computer activity 12 Including text in a worksheet

(a) Create the worksheet shown in Figure 8. Use a Title cell for the title 'Circles' and Text cells for the remaining text. Remember to click elsewhere on the worksheet or move position with the up or down keyboard arrow keys when you have finished entering each piece of text.

After evaluating a Maxima command, you will need to correctly position the horizontal cursor before inserting the next text cell.

(b) Use your worksheet to find the circumference and area of a circle of radius 5. (Remember to re-evaluate the worksheet, for example, by pressing `Ctrl-R`, after changing the value of `r`.)

2.9 Saving and printing your work

(Details on saving and printing your work when using the command-line interface are given in Subsection 10.5.)

After working with Maxima you will probably want to save your calculations so that you can reuse or read them at a later date. You can do this in wxMaxima using the `Save` or `Save As` options from the `File` menu.

There are various different formats in which you can save your work.

> It is recommended that you save your wxMaxima work as `The input without images` with the `.wxm` file type. This saves all your input but not any calculated output. You have to recalculate the output when you re-open the file, as demonstrated in Computer activity 13.

The alternative file type available is the `Whole document` with the `.wxmx` file type. This saves all the input and the output. Note that when you load a worksheet previously saved as a `.wxmx` file, although both the saved input and output are displayed, the input has not been evaluated by Maxima, so you cannot immediately use any results of the worksheet in new calculations. To re-evaluate the worksheet, and so recalculate all the results it contains, select `Evaluate All Visible Cells` from the `Cell` menu, or press `Ctrl-R`.

Input commands can also be exported to a `Maxima batch file`, with the `.mac` file type, using the `Export` option of the `File` menu. This is for more experienced users of Maxima.

You can open a saved wxMaxima worksheet by using the `Open` option from the `File` menu, or by double-clicking on the saved file on your computer.

Computer activity 13 Saving your work

Save your current worksheet then reload it, by following these steps.

(a) Select `Save As` from the `File` menu.

(b) In the window that appears:

(i) Choose the folder in which to save your work. You might like to save it in the folder that you created in Computer activity 1.

(ii) Choose a name for the file in which the worksheet will be saved.

(iii) Make sure the file is saved as a `The input without images` with the `.wxm` file extension.

(On a Windows computer, make sure that `The input without images (*.wxm)` is selected from the 'Save as type' drop-down list.)

(iv) Click 'Save'.

(c) Close wxMaxima, by selecting `Exit` from the `File` menu, or pressing `Ctrl-Q`, or clicking the small '`x`' at the top right of the window. (On a Mac select `Quit wxMaxima` from the `wxMaxima` menu or hold down the Command key while pressing Q.)

(d) Open your worksheet in wxMaxima again, by either

(i) double-clicking on the file you just saved, or

(ii) following the steps below:

- Start wxMaxima again.
- Select `Open` from the `File` menu. Maxima will ask if you want to save your just opened, blank worksheet. Click 'No'.
- Find the file containing your work and click 'Open'.

(e) Re-evaluate the worksheet by selecting `Evaluate All Visible Cells` from the `Cell` menu or pressing `Ctrl-R`.

> The calculations in the worksheet are evaluated and results displayed.

Printing is best achieved by first exporting the whole worksheet to a HTML document (a web page) by selecting `Export` from the `File` menu, then opening the HTML document using a web browser and printing from that. The worksheet can also be exported as source code for the LaTeX mathematical typesetting system using the same menu items. (Currently, these exports are not perfect representations of the worksheet.)

Alternatively, you can obtain an image of the entire wxMaxima window by taking a 'screenshot'. To do this on a Microsoft Windows computer, first click on the wxMaxima window, then press `Alt-PrintScreen` (that is, hold down the Alt keyboard key while pressing the `PrintScreen` key, which may be labelled PrtScr, Prt Scrn or something similar). This stores the image, which you can then paste into a suitable application. If you are using Windows Vista, Windows 7, 8 or 10, you might like to use the 'Snipping Tool', available by typing 'snip' into the Start menu. To take a screenshot on an Apple Mac computer, press `Cmd-Ctrl-Shift-4` (that is, hold down the Command, Ctrl and ⇑ keyboard keys while pressing the 4 key). Next press the space bar, move the camera pointer over the wxMaxima window and click. This stores the image, which you can then paste into a suitable application. You may also like to use the Grab application from the Utilities folder.

You can also paste individual output lines (or parts of output lines) from the worksheet into, for example, word-processed documents. Select the part of the output that you wish to paste, then from the menu obtained by clicking the right mouse button, select one of the following options.

- `Copy` – This copies the Maxima syntax for the selection. It can be useful if you wish to use the Maxima syntax in another document, or use the copied output as input to another Maxima command.

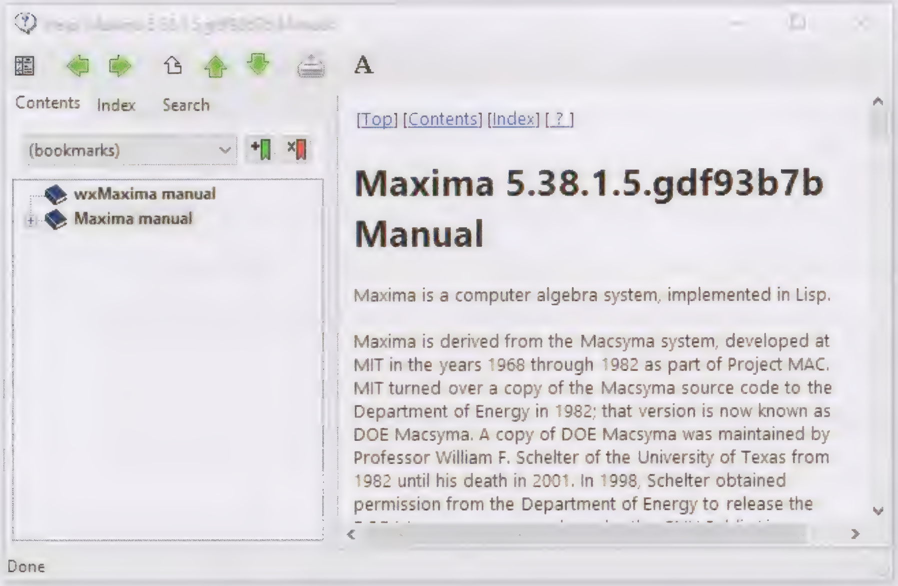
- `Copy as LaTeX` – This copies a LaTeX representation of the selection that can be pasted into a source file of the LaTeX mathematical typesetting system.

- `Copy as MathML` – This copies the selection as MathML, which is an international standard for the representation of mathematics. The MathML can be pasted into, for example, versions of Word from 2007 onwards, where it will be pasted as mathematics that can be edited using the ribbon maths editor. It can also be imported into LibreOffice documents, by, within LibreOffice, first creating a maths object using the `Insert > Object > Formula` menu options, then importing using `Tools > Import MathML from Clipboard`.
- `Copy as Image` – This copies the selection as an image that can be pasted into another document.

Whole cells can also be cut and pasted using the above options, by selecting the cell marker before clicking the right mouse button. However, when doing this, the LaTeX source contains a significant amount of formatting information that you may not require, and the MathML option for a whole cell does not work in the version of wxMaxima current at the time of writing.

2.10 Getting help

There are several ways in which you can obtain help with using Maxima, beyond this *Guide*. In wxMaxima, you can access the complete Maxima manual by selecting `Maxima Help` from the `Help` menu. This opens the window shown in Figure 9. You can use the Contents, Index and Search tabs to look for the information that you want.



Maxima 5.38.1.5.gdf93b7b Manual

Maxima is a computer algebra system, implemented in Lisp.

Maxima is derived from the Macsyma system, developed at MIT in the years 1968 through 1982 as part of Project MAC. MIT turned over a copy of the Macsyma source code to the Department of Energy in 1982; that version is now known as DOE Macsyma. A copy of DOE Macsyma was maintained by Professor William F. Schelter of the University of Texas from 1982 until his death in 2001. In 1998, Schelter obtained permission from the Department of Energy to release the

Figure 9 The wxMaxima help system

Alternatively, Maxima also includes commands that you can use to obtain help. These are demonstrated in the following activity and summarised in the table that follows.

Computer activity 14 Getting help!

(a) Enter

```
? float;
```

to show the help for the `float` command.

Note the space between the `?` and `float`, which is needed. This is one of the few times in Maxima when spaces matter!

The '`?`' is the Maxima syntax to request help on the command given.

The help information for the command may include more detail than you need!

After giving the help information, Maxima displays the output `true`. This means that the command you entered was successful.

(b) Suppose that you cannot remember the Maxima command for square root, but you do remember that it was something like *sqr*.

Enter

```
?? sqr;
```

to list all the help information titles that contain the letter sequence 'sqr'.

A list of possible titles is displayed.

Title number 1 is what you were looking for, so enter `1;` to display the relevant information.

If more than one title is displayed in response to a `??` command, then Maxima will not continue until you enter one of the following:

- the appropriate number, to display your chosen information
- several numbers, separated by spaces, to see help information on several topics
- `all` to display the information on all topics listed, or
- `none` to see no information.

A box will be shown around the cell marker while Maxima is waiting for your response.

Help commands

Operation	Command	Example

Get help on a command	`? command`
	or `describe(command)`
		`? float;`
		`describe(float);`

Find help information whose title contains the given text	`?? ▨`
	or `describe(▨, inexact)`
		`?? flo;`
		`describe(flo, inexact);`

Note 1: the space after the `?` is needed.

Note 2: the second pair of commands displays a list of the titles of all help information pages whose title contains the given text. To obtain help on a particular topic listed, type the number corresponding to the required title in the list, followed by a semicolon.

There is further help available in the Frequently asked questions (FAQs) section of the OU Maxima website: `learn1.open.ac.uk/site/maxima`.

3 Using Maxima

In Section 2 of this *Guide* you revised how to use wxMaxima and how to perform numerical calculations. Here, you will revise how to use Maxima for algebraic manipulations, equation solving, calculus and graph plotting.

If you are familiar with Maxima from MST124, you should be able to quickly skim through this section and try some of the activities to check your skills, revising them more thoroughly as necessary. If you require further details on any of the topics covered, you might like to read the appropriate section of the MST124 *Computer algebra guide* available on the OU Maxima website at `learn1.open.ac.uk/site/maxima`.

3.1 Algebra and functions

You can enter algebraic expressions into Maxima in a similar way to numerical expressions, using the operations and functions listed in Subsection 2.1. You can assign algebraic expressions to variables, using the assign (`:`) command and you can also define your own functions using the `:=` operator. These are demonstrated in the following activity.

Computer activity 15 Defining algebraic variables and functions

(a) Define the variable `p` in Maxima to be the algebraic expression

$$2x + \sqrt{1 + 3x^6}$$

(b) Find the value of `p` when $x = 3$ by entering

```
subst(3,x,p);
```

The command name `subst` is short for 'substitute'.

The command `subst(3,x,p);` substitutes the value 3 for the variable `x` in the expression `p`.

(c) Define the function f in Maxima, with the rule $f(x) = \dfrac{x^4+x+1}{x^3-13x+12}$, by entering the following command. (Use the characters `:` followed by `=` to type the `:=` operator.)

```
f(x):=(x^4+x+1)/(x^3-13*x+12);
```

(d) Calculate the value of $f(2)$, by entering `f(2);`

(e) Try to calculate the value of $f(3)$.

This gives an error.

This is because when Maxima tries to evaluate $f(3)$, it finds that this involves dividing by zero, since the denominator of f is zero when $x = 3$, so it returns an error message. The function f is undefined at $x = 3$.

(f) Enter `g(1);`

Maxima returns the expression `g(1)`, unevaluated.

Since the function `g` has not been defined, Maxima cannot calculate a value. It knows only that the value is `g(1)`.

(g) Enter `functions;`

This displays the value of the system variable `functions`, which is a list of all your currently defined functions in this Maxima session.

Notice the difference between defining a *function* using `:=` and assigning an expression to a *variable* using `:`. A function has an argument and can be evaluated at different values of the argument, while a variable does not.

Once you have defined a function in Maxima it is remembered by the system for the rest of your session, unless you remove it using the `kill` command that you revised in Subsection 2.6 of this *Guide*.

Substituting into an expression

Operation	Command	Example
Substitute	`subst( value , variable , expression )`	`subst(4, x, x^2+1);` which substitutes 4 for `x` in `x^2+1`.

Note: `expression` indicates that the argument of the command must be an expression, or a variable whose value is an expression.

Working with functions

Operation	Command	Example
Define a function	`:=`	`f(x):=2*x+3;`
Evaluate a function at a value	`function( )`	`f(1);`
List all user-defined functions	`functions`	`functions;`
Remove a defined function	`kill( function )`	`kill(f);`

Maxima can also re-express algebraic expressions in different, but equivalent forms. It is hard for a computer to automatically simplify algebraic expressions, because often there is no single 'simplest form' for a particular expression, and the computer has no way of knowing which form you prefer.

Instead, a range of commands are provided (which are listed in the following table) which you can use to instruct Maxima to attempt to re-express an expression in different ways. You can use these commands to simplify an expression in the way you want.

Algebraic and trigonometric manipulation commands

Operation	Command	Example
Expand brackets	`expand( )`	`expand((x+1)^2);`
Factorise	`factor( )`	`factor(2*x+4*x^2);`
Simplify	`fullratsimp( )`	`fullratsimp((2*x+4*x^2)/x);`
Simplify something involving exponentials and logarithms	`radcan( )`	`radcan(log(x^2));`
Combine logarithms	`logcontract( )`	`logcontract(log(a)+log(b));`
Expand trigonometric functions of sums, differences and multiples of angles	`trigexpand( )`	`trigexpand(sin(A+B));`
Express powers of sines and cosines in terms of sines and cosines of multiple angles	`trigreduce( )`	`trigreduce(sin(x)^2);`
Simplify trigonometric expressions	`trigsimp( )`	`trigsimp(sin(x)^2+cos(x)^2);`
Simplify algebraic fractions containing trigonometric functions	`trigrat( )`	`trigrat((sin(2x))/cos(x));`

Computer activity 16 Rearranging algebraic expressions

(a) Enter

```
expand( (1+sin(x))*(x+4)^2 );
```

to expand the expression $(1+\sin x)(x+4)^2$.

The command `expand` multiplies out the expression that is its argument. Spaces have been included simply to help with the reading of the command.

(b) Enter

```
factor( 3*x^3+11*x^2+8*x-4 );
```

to factorise the expression $3x^3+11x^2+8x-4$.

The command `factor` factorises, where possible, the expression that is its argument.

(c) Enter

```
fullratsimp( (2*x^3-3*x^2+1)/(x-1) );
```

to simplify the expression $\dfrac{2x^3-3x^2+1}{x-1}$.

The command `fullratsimp` simplifies, where possible, the expression that is its argument. It does this by using techniques for multiplying out brackets, collecting terms over a common denominator and cancelling out common factors. Simplifications requiring other techniques may not be handled by `fullratsimp` in the way you would expect.

The name `fullratsimp` is short for 'full rational simplify'.

(d) Enter

```
radcan( log(x^3-3*x-2)-2*log(x+1) );
```

to simplify the expression $\ln(x^3-3x-2)-2\ln(x+1)$.

The command `radcan` uses the rules of indices and logarithms to simplify, if possible, an expression containing roots (for example square roots or cube roots), powers and logarithms.

Note that roots are often known as *radicals*. The command `radcan` converts an expression involving radicals (and related functions such as powers and logarithms) into a canonical (or standard) form.

(e) Enter

```
logcontract( log(2*x)-2*log(x)+log(y) );
```

to simplify the expression $\ln(2x) - 2\ln x + \ln y$.

The command `logcontract` combines, where possible, several terms involving logarithms.

(f) Enter

```
trigexpand( tan(3*x) );
```

The `trigexpand` command attempts to express trigonometric functions of sums, differences and multiples of angles in terms of powers of trigonometric functions of individual angles.

(g) Enter

```
trigreduce( cos(x)^4 );
```

The `trigreduce` command attempts to express powers of sines and cosines of angles in terms of sines and cosines of multiples of the angles. It *reduces* the powers of the functions.

(h) Enter

```
trigsimp( cos(x)^2*(1+tan(x)^2) );
```

to simplify $\cos^2 x\left(1 + \tan^2(x)\right)$.

The `trigsimp` command attempts to simplify expressions involving trigonometric functions using the fact that $\sin^2 x + \cos^2 x = 1$.

(i) Enter

```
trigrat( tan(x)/sin(2*x) );
```

to simplify $\dfrac{\tan x}{\sin(2x)}$.

The `trigrat` command attempts to simplify algebraic fractions involving trigonometric functions.

You can also access many of these commands for manipulating expressions from the wxMaxima menus. For example, to factorise an expression you can first enter the expression in the worksheet, select it by highlighting the text using the mouse or clicking on the corresponding output line number, then select `Factor Expression` from the `Simplify` menu (or from the menu obtained by right clicking on a highlighted output line number).

Other options on these menus, such as `Substitute...` open a window for you to enter the command options. Options that work in this way are identified by `...` at the end of the option name.

3.2 Plotting graphs

You can plot graphs of expressions or functions in wxMaxima using the `wxplot2d` command. This command inserts the graph into your worksheet, as demonstrated in the following activity. The '`2d`' in the name of this command comes from the fact that its graphs are two-dimensional plots. The '`wx`' is because it is a wxMaxima command.

If you are using another interface to Maxima, you should use the command `plot2d` instead. You use it in the same way as `wxplot2d`, but it will plot the graph in a different computer window, which you then need to close before you can create another plot.

Maxima plots graphs of functions by calculating a large number of points on the graph and joining them up.

Computer activity 17 Plotting graphs

(a) Plot the graph of the function $x \sin x$ for values of x such that $-5 \leq x \leq 5$, using the command

```
wxplot2d(x*sin(x), [x,-5,5]);
```

The first argument of the `wxplot2d` command is the expression to be plotted, and the second argument specifies the variable in the expression together with the least and greatest values that you want it to take in the graph. These are grouped in a list.

Note that wxMaxima displays the graph next to a line label beginning `%t`, such as `(%t1)`. This is how wxMaxima shows *intermediate* output, that is, output other than the final result of the command. The `wxplot2d` command has no final result, just intermediate output, which is the graph.

(b) Define the function f with the rule

$$f(t) = \frac{t^2 + 2t + 3}{t^2 + 2t - 3}$$

and plot the graph of f for values of t such that $0 \leq t \leq 3$, using the command

```
wxplot2d(f(t), [t,0,3]);
```

Notice that the first argument of the `wxplot2d` command must be $f(t)$ rather than just f and that since the independent variable is t, a range of values of t is given as the second argument.

The graph produced shows a large spike near $t = 1$. This is because $f(t)$ is not defined at $t = 1$, since its denominator is zero there.

To plot the graph, Maxima has evaluated f at values of t in the range $0 \leq t \leq 3$, obtained large values near $t = 1$ and joined up the corresponding points.

To obtain a better graph, you should restrict the range of the vertical axis. You are asked to do this next.

(c) Plot the graph of f over the range $0 \leq t \leq 3$, with the vertical range of the graph restricted to $-10 \leq y \leq 10$ using the following command.

```
wxplot2d(f(t), [t,0,3], [y,-10,10]);
```

The range of the vertical axis is restricted by including an additional argument which is a list starting with the *keyword* `y` and followed by the minimum and maximum values to be shown on the axis.

A warning:

```
plot2d:  some values were clipped.
```

is given. This is because some of the points on the graph with x-coordinates between 0 and 3 could not be displayed due to the range of values of y specified.

Warning

When plotting graphs, do not assign the expression to be plotted (or any other expression) to the *variable* `y` as this will cause an error when the keyword `y` is used in the specification of the range of the y-axis. Similarly, the variable in the expression to be plotted should not have been previously assigned to a specific value. (Remember, a variable with an assigned value can be deleted using the `kill` command.)

It is, however, possible to assign the expression to be plotted to the *function* `y`.

You can also plot multiple graphs in the same diagram by including a list of expressions as the first argument of the command, as shown in the following activity.

Computer activity 18 Plotting multiple graphs

(a) Define the functions p and q to be

$$p(x) = x\cos x \qquad \text{and} \qquad q(x) = x - \frac{x^3}{2}.$$

(b) Plot graphs of the two functions over the range $-3 \leq x \leq 3$ on the same diagram by entering:

```
wxplot2d([p(x),q(x)],[x,-3,3]);
```

Notice that the first argument of the command `wxplot2d` is a list containing the two functions to be plotted.

The curves are automatically plotted in different colours, and a *legend* showing which function corresponds to which curve is included at the top right-hand corner of the graph.

The different ways in which you can use the `wxplot2d` command are summarised below.

Plotting graphs

Operation	Command	Example

Plot a graph	`wxplot2d(expression, horizontal range,...)`	`wxplot2d(x^2,[x,0,1]);`
		which plots the graph of x^2 for x between 0 and 1
		`wxplot2d(x^2,[x,0,1],[y,0,2]);`
		which plots the graph of x^2 for x between 0 and 1, and with vertical axis values between 0 and 2

Plot several graphs	`wxplot2d(list of expressions, horizontal range,...)`	`wxplot2d([x^2, 2*x],[x,0,1]);` which plots graphs of x^2 and $2x$ for x between 0 and 1

Changing graph proprieties

Maxima lets you control many properties of the graph, for example the colour of the curves plotted and the text in the legend. This is done by including additional arguments in the `wxplot2d` command, each being a list beginning with a keyword describing the property to be changed and followed by the required value of the property. Where properties apply to individual curves, a value should be listed for each curve plotted. This is demonstrated in the following activity.

Computer activity 19 Changing the properties of a graph

Plot the graph of the functions p and q from Computer activity 18 again, this time with

- p coloured black and q green
- the graph legend changed to refer to the names p and q rather than the expressions they represent
- the vertical axis labelled y

by entering

```
wxplot2d([p(x),q(x)], [x,-3,3], [color, black, green],
[legend, "p", "q"], [ylabel, "y"]);
```

The additional settings are given as lists, each beginning with a keyword.

The colours of the curves are given in a list beginning `color` (American spelling), and one colour is listed for each curve. Maxima uses the first colour listed for the graph of the first expression listed and so on. If you do not include enough colours, Maxima starts using the listed colours again.

The list beginning `legend` specifies the legend text for each curve, each enclosed in double quote marks.

The list beginning `ylabel` sets the text to be displayed on the vertical axis. If required, text for the horizontal axis label can be specified using a list beginning with the keyword `xlabel`.

The legend can be moved to the bottom right-hand corner of the graph by including the additional argument

```
[gnuplot_preamble, "set key bottom"]
```

(Gnuplot is the underlying software that Maxima uses to plot graphs.)

The legend can be suppressed by including the argument

```
[legend, false]
```

Changing the graph size

Finally, the size of the plot within your worksheet can be changed using the `wxplot_size` system variable. For example, entering

```
wxplot_size:[800,600];
```

changes the size of subsequently plotted graphs from the default setting of 600 × 400 pixels to 800 × 600 pixels.

Summary

Graph plotting options

Option	Argument	Example
Vertical range	`[y, ■, ■]`	`[y, 0, 5]`
Curve colour	`[color, ■, ...]`	`[color, red]`
Legend text	`[legend, "■", ...]`	`[legend, "Car A"]`
Turn legend off	`[legend, false]`	`[legend, false]`
Position legend at the bottom of the graph	`[gnuplot_preamble, "set key bottom"]`	`[gnuplot_preamble, "set key bottom"]`
Position legend to the right of the graph	`[gnuplot_preamble, "set key outside"]`	`[gnuplot_preamble, "set key outside"]`
Horizontal axis label	`[xlabel, "■"]`	`[xlabel, "t (h)"]`
Vertical axis label	`[ylabel, "■"]`	`[ylabel, "s (km)"]`
Change the size of the graph	`wxplot_size:[■,■]`	`wxplot_size:[800,600];`

Note: for a full list of the possible options, view the Maxima help obtained using the `?? plotting options;` command.

Plotting curves represented by implicit equations

The `wxplot2d` command can only be used to plot curves represented by equations in *explicit* form; that is, written in the form $y = f(x)$, where f is a function.

To plot a curve represented by an equation in *implicit form*, such as $x^2 + y^2 = 1$, you need to use the `wximplicit_plot` command. This is demonstrated in Computer activity 20.

The `wximplicit_plot` command is not part of the basic Maxima collection of commands, so you have to add it by loading an additional Maxima **package**, the `implicit_plot` package, which was installed on your computer when you installed Maxima. The Maxima command to load an additional package is `load`, which is also demonstrated in Computer activity 20. Each package needs to be loaded only once per Maxima session.

If you are using a Maxima interface other than wxMaxima, then you should use the `implicit_plot` command instead of the `wximplicit_plot` command. You load and use it in exactly the same way as `wximplicit_plot`.

Computer activity 20 Plotting a circle

Plot the circle represented by the equation $x^2 + y^2 = 1$, as follows.

(a) Load the implicit_plot package by entering

```
load(implicit_plot);
```

Remember to type the underscore symbol (_) in the middle of the package name.

This package provides the wximplicit_plot command.

(b) Plot the circle by entering

```
wximplicit_plot(x^2+y^2=1, [x,-2,2], [y,-2,2]);
```

Again, remember to type the underscore symbol (_) in the middle of the command name.

The first argument of the command is the equation to be plotted. Notice that in Maxima, equations are given using the usual equals sign.

The second argument specifies the variable to be plotted on the horizontal axis, and the range of values of that variable to be plotted.

The third argument specifies the variable (and its range) to be plotted on the vertical axis. You must include this third argument.

The circle with equation $x^2 + y^2 = 1$ has centre $(0, 0)$ and radius 1. So the circle lies within $-2 \leq x \leq 2$ and $-2 \leq y \leq 2$ and hence these seem suitable ranges to use.

The curve that Maxima plots, however, does not look very circular! This is discussed below.

Note that the first argument of the wximplicit_plot command is an *equation*, whereas the first argument of the wxplot2d command is an *expression* (or a function or variable representing an expression). If an expression is given as the first argument, wximplicit_plot will plot the graph of the equation obtained by setting the expression equal to zero.

The curve that you plotted in Computer activity 20 does not look circular because it has different scales on the axes. Whatever ranges of x- and y-values you specify for a plot, by default Maxima makes the y-axis appear $\frac{3}{5}$ as long as the x-axis.

To specify the use of equal scales, you can add the following additional argument to the wximplicit_plot command:

```
same_xy
```

Note that in some older releases of Maxima, the option needed to specify equal scalings on coordinate axes was

```
[gnuplot_preamble, "set size ratio -1"]
```

If the same_xy option does not change the scales of your graph, then use this option instead.

Computer activity 21 Plotting a circular circle!

Plot the circle $x^2 + y^2 = 1$, using equal scales on the x- and y-axes by entering

```
wximplicit_plot(x^2+y^2=1, [x,-2,2], [y,-2,2], same_xy);
```

Including the same_xy (or gnuplot_preamble) argument in the wximplicit_plot command can be tiresome if you are plotting lots of curves and need equally scaled x- and y-axes for each. An alternative is to change the default behaviour of Maxima so that it uses axes with equal scales for the rest of your session. You will see how to do this in the following activity.

Computer activity 22 Plotting another circular circle!

Plot the circle $(x - 1)^2 + y^2 = 3$, using equal scales on the x- and y-axes as follows.

(a) Tell Maxima to always use equal scales on the x- and y-axes by entering

```
set_plot_option(same_xy)$
```

This sets same_xy to be a default option for all plots during the rest of your Maxima session. The line ends with $ to prevent the display of the output of the command. Without this, the output consists of a long list of all the current default settings, which can safely be ignored.

If you are using an older release of Maxima that uses the gnuplot_preamble option, use

```
set_plot_option([gnuplot_preamble, "set size ratio -1"])$
```

instead.

(b) Plot the circle by entering

```
wximplicit_plot((x-1)^2+y^2=3, [x,-1,3], [y,-2,2]);
```

If you wish to stop plotting graphs with equal scales and restore the default Maxima plotting behaviour, then set the same_xy plotting option to be false using

```
set_plot_option([same_xy, false])$
```

If you are using an older version of Maxima, the command needed to reset the default view is

```
set_plot_option([gnuplot_preamble, ""])$
```

To plot more than one curve at once you can list their equations within square brackets as the first argument of wximplicit_plot, as you did with the wxplot2d command. The equations do not necessarily have to be in implicit form. This is demonstrated in the following activity.

Computer activity 23 Plotting a parabola and a circle

Plot the parabola $y = x^2 - 6x + 9$ and the circle $(x-3)^2 + (y-2)^2 = 3$ over the range $0 \leq x \leq 6$, $0 \leq y \leq 6$ by entering

```
wximplicit_plot([y=x^2-6*x+9, (x-3)^2+(y-2)^2=3],
[x,0,6], [y,0,6]);
```

The commands introduced in this section are summarised below.

Loading additional packages

Operation	Command	Example
Load a package	load(package name)	load(implicit_plot);

Plotting curves represented by equations in implicit form

Operation	Command	Example
Plot a curve represented by an equation in implicit form	wximplicit_plot(equation , horizontal range , vertical range , ...)	wximplicit_plot(x^2+y^2=1, [x,-1,1], [y,-1,1]);
Plot several curves represented by equations in implicit form	wximplicit_plot(list of equations , horizontal range , vertical range , ...)	wximplicit_plot([x^2+2*y^2=4, y=x^3], [x,-1,1], [y,-1,1]);

Note 1: the implicit_plot package needs to be loaded before this command can be used.

Note 2: the first argument is an equation or a list of equations. Alternatively it can be an expression, or a list of expressions, in which case the equation formed by setting the expression equal to zero is plotted.

Note 3: equations in explicit form can also be plotted using this command.

Graph plotting options

Option	Argument	Example

Set axes to have equal scales
(in older versions of Maxima)
	`same_xy`
	`[gnuplot_preamble, "set size ratio -1"]`
		`same_xy`
		`[gnuplot_preamble, "set size ratio -1"]`

Setting default plotting options

Operation	Command	Example
Set the default plotting options	`set_plot_option( option )`	`set_plot_option(same_xy)`

3.3 Solving equations

You can use the Maxima `solve` command to find the exact solution(s) of equations, and to change the subject of an equation, as demonstrated in the following activity.

Remember, to type an equation in Maxima you use an equals sign in the usual way. For example, you can type `3*x+2=4`. You can also assign equations to a variable in a similar way to expressions. For example, to assign the equation $3x + 2 = 4$ to the variable `q`, enter

```
q: 3*x+2=4;
```

Computer activity 24 Solving and rearranging equations

(a) Solve the equation $x^3 - 4x^2 + 2x + 4 = 0$ by entering

```
solve( x^3-4*x^2+2*x+4=0 );
```

This is a *cubic* equation, since the highest power of x is 3. Each cubic equation has at most three solutions.

Notice that Maxima displays the solutions of the equation as a list.

(b) Assign the equation $y = \ln(ux + v)$ to the variable `eqn`.

Remember, the Maxima function for ln is `log`.

(c) Rearrange the equation to make x the subject by entering

```
solve(eqn, x);
```

The second argument of the `solve` command gives the variable we wish to 'solve' for.

(d) Solve the equation $\tan(3x - 1) = \sqrt{3}$.

When solving equations involving trigonometric functions Maxima will only try to find one solution, even though many others may exist.

The warning

```
solve: using arc-trig functions to get a solution.
Some solutions will be lost.
```

is given to alert you to this.

(e) Solve the equation $x^2 - 4x + 20 = 0$.

The solutions are given in terms of `%i`.

This is how Maxima represents the imaginary number i whose square is equal to -1. There are no solutions to this equation that are *real numbers*, but Maxima has found solutions that are *complex numbers*.

(f) Try to use Maxima to solve the equation $x^9 + 2x - 1 = 0$.

Here, Maxima returns (within a list) the equation, slightly rearranged, but unsolved.

The `solve` command always tries to find an exact solution of the equation using algebraic manipulations. If it is unable to do this, then it returns the original equation, as here.

You will revise later in this section how to find approximate numerical solutions of such equations.

The use of the `solve` command illustrated in the previous activity is summarised below.

Solving and changing the subject of an equation

Operation	Command	Example
Solve an equation, exactly	`solve(equation)`	`solve(2*x^2-1=0);`
Solve an equation for a variable	`solve(equation, variable)`	`solve(2*a*b-3*b=0, a);`

You saw in Computer activity 24 that the output of the `solve` command is a *list* of equations. To extract the individual solutions you need to be able to manipulate lists. This is demonstrated in the following activity.

Computer activity 25 Working with lists

(a) Assign the solutions of the equation $x^2 - x - 1 = 0$ to the variable `solns` by entering

```
solns:solve(x^2-x-1=0);
```

Remember, the solutions of an equation are given as a list, so `solns` is a list.

(b) Display the first element of the list `solns` by entering

```
solns[1];
```

The individual elements of a list can be obtained by adding the element number (which is known as the `index` of the element) after the name of the list and within square brackets.

In this case, the first element of the list is the equation $\texttt{x=-}\frac{\sqrt{5}-1}{2}$. The *value* of the first solution of the original equation is the right-hand side of this.

(c) Display the exact value of the first solution of $x^2 - x - 1 = 0$ by entering the command

```
rhs(solns[1]);
```

The command `rhs` gives the right-hand side of an equation.

(d) Display the exact value of the second solution of the equation similarly.

Lists

Operation	Command	Example
Extract an element of a list	`list[index]`	`A[2];`

Extracting the left- and right-hand sides of an equation

Operation	Command	Example
Left-hand side of an equation	`lhs(equation)`	`lhs(4*x+1=2*x-2);`
Right-hand side of an equation	`rhs(equation)`	`rhs(4*x+1=2*x-2);`

Another use of the `solve` command is to solve two or more simultaneous equations. To use the command in this way, you must enter the equations as a list, and enter the variables to be solved for as another list, as indicated below.

Solving simultaneous equations

Operation	Command	Example
Solve simultaneous equations	`solve(list of equations, list of variables)`	`solve([2*x+y=4, x-2*y=1], [x,y]);`

Computer activity 26 Solving simultaneous equations

(a) Use Maxima to solve the simultaneous equations

$$3x + 2y = 1$$
$$5x + 3y = 3$$

as described below.

(i) For simplicity, assign the first equation to the variable `eq1` and the second equation to the variable `eq2`.

> Maxima might reorder the terms of the equations when displaying the output.

(ii) Solve the pair of equations (`eq1` and `eq2`) for the two unknowns (x and y) by entering

```
solve( [eq1,eq2], [x,y] );
```

> Maxima's output is `[[x=3, y=-4]]`
>
> The solution list contains the element `[x=3, y=-4]` which is itself a list of the pair of values of `x` and `y` that solve the equations.

(b) Solve the equations

$$y = x^2 - 6x + 9$$
$$(x-3)^2 + (y-2)^2 = 3$$

which gives the points of intersection of the parabola and circle plotted in Computer activity 23.

> Notice there are four solutions – Maxima gives a list of each pair of values of x and y which satisfy the equations.

In Computer activity 24(f) you saw an example where the `solve` command failed to find the exact solution of an equation. In such circumstances you can use another Maxima command, `find_root`, to try to find an approximate solution.

Whereas the `solve` command manipulates the equation mathematically to find a solution, the `find_root` command performs a series of numerical calculations to find a decimal approximation to a solution. This is known as solving the equation **numerically**, or finding a **numerical approximation** to a solution.

The solutions of an equation of the form $f(x) = 0$ are the x-coordinates of the points at which the graph of $y = f(x)$ crosses the x-axis. If f is a continuous function and $[a, b]$ is an interval included in its domain such that one of $f(a)$ and $f(b)$ is negative and the other is positive, then the graph of $f(x)$ must cross the x-axis at one or more points in the interval (a, b), as illustrated in Figure 10. So there must be at least one solution of the equation $f(x) = 0$ in the interval (a, b).

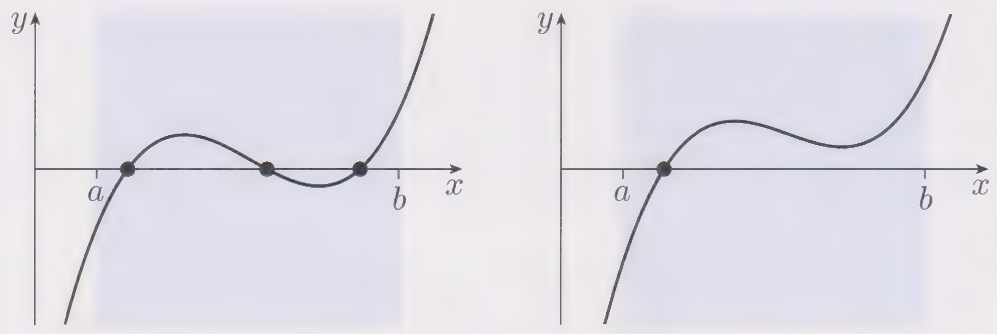


Figure 10 There is at least one solution of $f(x) = 0$ in the interval (a, b) if $f(a)$ and $f(b)$ have different signs

The `find_root` command looks for a single solution of an equation of the form $f(x) = 0$ that lies within an interval that you specify. The command requires that the values of f at each endpoint of the interval have different signs. If this is not the case, then Maxima gives an error message.

Solving an equation numerically

Operation	Command	Example

Solve an equation numerically	`find_root( expression ,`
	`variable ,`
	`interval start value ,`
	`interval end value )`
		`find_root(2*x^2-1,x,0,1);`

Note 1: the command finds a value of the variable at which the expression is zero.

Note 2: the values that the expression takes at the endpoints of the interval must have different signs.

Computer activity 27 Solving equations numerically

(a) Find a numerical approximation to a solution of the equation $x^9 + 2x - 1 = 0$ seen in Computer activity 24(f) as follows.

(i) First, define the function f in Maxima, with the rule $f(x) = x^9 + 2x - 1$.

(ii) Next, plot the graph of $f(x)$ over the interval $-1 \leq x \leq 1$, say. This will help you to find an interval that contains a solution.

From the graph, you can see that $f(0) < 0$ and $f(1) > 0$, so there is a solution in the interval $(0, 1)$.

If there were no solution in the interval over which you plotted the function, then you could plot the function over a larger interval, and so on, until you find a suitable interval, if one exists.

(iii) Find a numerical approximation to the solution of the equation in the interval $(0, 1)$ by entering

```
find_root(f(x), x, 0, 1);
```

Remember to type the underscore symbol (_) in the middle of the command name.

Notice that the first argument of the `find_root` command is an *expression* (or a function defined by an expression) not an *equation*. Maxima finds an input value at which the expression takes the value 0.

(b) Using similar methods, find a solution of the equation $\cos x = \sqrt{x}$.

Hint: consider the function $g(x) = \cos x - \sqrt{x}$.

3.4 Calculus

The Maxima commands for differentiating and integrating expressions are `diff` and `integrate`. These are demonstrated in the following activity.

Note that sometimes Maxima may give the result of differentiating or integrating an expression in a different, but equivalent, algebraic form to the result that you might obtain when differentiating or integrating by hand. If this happens, then you might like to check that the two answers are equivalent. You can do this by finding the difference between the two answers and checking that it simplifies to zero (in the case of differentiation) or a constant (in the case of integration).

A rougher check is to use Maxima to plot graphs of the two answers and check they appear to be the same (for differentiation) or vertical translations of each other (for integration).

Computer activity 28 Differentiating and integrating

(a) Find the derivative of $f(x) = \dfrac{\cos x}{x^2+3}$, by entering

```
diff(cos(x)/(x^2+3), x);
```

The first argument of the `diff` command is the expression to be differentiated, and the second argument is the variable to differentiate with respect to.

You must include the second argument. If you do not, then the answer returned by Maxima will be the derivative multiplied by `del(x)`, which represents the differential of `x`.

As you would expect, the first argument can also be a variable that you have previously defined to be an expression, or a function you have previously defined.

(b) Find the second derivative of $g(u) = u^2 \sin u$, by entering

```
diff(u^2*sin(u), u, 2);
```

The third argument of `diff` specifies the number of times that the expression is to be differentiated. (If this is omitted, as in part (a), then the first derivative is calculated.)

(c) Find the third derivative of $h(t) = \dfrac{\ln t}{t^2}$.

Remember, the Maxima function for ln is `log`.

(d) Evaluate $\int x \sin^2(x)\,dx$ by entering

```
integrate(x*sin(x)^2, x);
```

The first argument of `integrate` is the expression to integrate, and the second is the variable to integrate with respect to.

The output returned by Maxima does not include an arbitrary constant. That is, Maxima gives an antiderivative of the expression, not its indefinite integral. When you write down an indefinite integral that you have found using Maxima, you need to remember to add an arbitrary constant yourself.

(e) Evaluate the definite integral $\int_0^1 t^2 e^t \, dt$ by entering

```
integrate(t^2*%e^t, t, 0, 1);
```

The third and fourth arguments of `integrate` are the lower and upper limits of integration, respectively.

Notice that, as usual, Maxima returns an exact result. Use the `float` command to find a decimal approximation, if required.

(f) Evaluate the definite integral $\int_1^2 \frac{1}{x^2} \, dx$

(g) Try to evaluate $\int_0^1 \tan(e^{-x}) \, dx$

Maxima cannot calculate this integral, so it is displayed unevaluated.

As you saw in Computer activity 28(g) it is not always possible to express the result of an integral exactly. Sometimes, Maxima may give a result in terms of the so-called *special functions*. These cannot be simply expressed in terms of simple functions, but many can be defined in terms of antiderivatives of them. Such functions include `erf`, which is known as the *Error function*, and `gamma_incomplete`, the *incomplete gamma function*.

You will revise how to calculate numerical approximations to definite integrals that cannot be calculated symbolically later in this section.

The `diff` and `integrate` commands are summarised below.

Calculus

Operation	Command	Example
Differentiate	`diff(□, variable)`	`diff(sin(x^2),x);`
Differentiate multiple times	`diff(□, variable, positive integer)`	`diff(log(x),x,3);` to differentiate three times
Integrate (find an antiderivative)	`integrate(□, variable)`	`integrate(x^2,x);`
Evaluate a definite integral	`integrate(□, variable, lower limit, upper limit)`	`integrate(sin(x),x,0,1);`

If you wish to assign the result of a `diff` or `integrate` command to a named function, then you need to take care that the function is defined to be equal to the *result* of the operation, not the operation of differentiating or integrating itself.

This is demonstrated in the following activity.

Computer activity 29 Defining a function to be the derivative or integral of another function

(a) Define the function $f(x) = x\sqrt{1+4x^2}$.

(b) Try to define `df(x)` to be the derivative of $f(x)$, by entering

```
df(x):=diff(f(x),x);
```

(c) Try to find the value of the derivative of f at $x = 3$ by evaluating `df(x)` at $x = 3$; that is, by entering

```
df(3);
```

This gives an error. The problem is that when you entered the command

```
df(x):=diff(f(x),x);
```

Maxima did not perform the differentiation. Instead, it just noted that `df(x)` means `diff(f(x),x)`. Then, when you entered `df(3)`, Maxima understood this to mean `diff(f(3),3)`, which is meaningless, as you cannot differentiate with respect to a particular number.

To correctly define `df(x)` to be the *result* of differentiating `f(x)`, you need to force Maxima to perform the differentiation at the point `df(x)` is defined.

You can do this using the syntax `''(`▨`)` (which contains two single quotation marks, not a double quotation mark, and a pair of round brackets). This is demonstrated below.

(d) Define `df(x)` to be the *result* of differentiating f with respect to x, by entering

```
df(x):=''(diff(f(x),x));
```

Remember that `''` is two *separate* quotation marks. Use the upright quotation mark on your keyboard (`'`), not the sloping one (`` ` ``). Don't forget to include the outer round brackets too.

Notice that this time the output line shows that `df(x)` is indeed defined to be the result of differentiating `f(x)`.

(e) Find the value of the derivative of f at $x = 3$ by entering

```
df(3);
```

(f) Use a similar approach to define `F(x)` to be an antiderivative of `f(x)`.

(g) Calculate `F(1)-F(0)` and confirm this is the same value as $\int_0^1 f(x)\,\mathrm{d}x$.

The command used to force the evaluation of another command is summarised below.

Forcing the evaluation of a command

Operation	Command	Example
Force the evaluation of a command	`''(▨)`	`g(x):=''(diff(x^4,x));`

Assumptions

Sometimes you may want to integrate a function that contains a variable other than the one you are integrating with respect to. For example, the integral $\int x^n\,\mathrm{d}x$ contains the variable n as well as the variable x. When you use Maxima to find such an integral, you sometimes need to provide information about the other variable as shown in the next activity.

Computer activity 30 Giving Maxima more information

(a) Try to use Maxima to find the integral $\int x^n\,\mathrm{d}x$, as follows.

(i) First, ensure that `n` has no predefined value, by entering

```
kill(n);
```

(ii) Then, enter

```
integrate(x^n, x);
```

Maxima asks the question: `Is n equal to -1?`

This is because the answer to $\int x^n\,\mathrm{d}x$ depends on the value of n.

If $n = -1$, then the integral is $\int \frac{1}{x}\,\mathrm{d}x$, which is equal to $\ln|x| + c$, whereas if n is any other value, then the integral is equal to $\frac{x^{n+1}}{n+1} + c$. So Maxima cannot calculate the integral without knowing more about n.

Enter the answer `n;`
(which is short for `no`)

You could also answer the question with `y;` to mean `yes`.

Maxima displays the expected result.

(b) An alternative approach is to give Maxima information about n *before* you use the `integrate` command. You can do this as follows.

(i) Tell Maxima that $n \neq -1$ by entering

```
assume(notequal(n,-1));
```

The `assume` command tells Maxima information about variables. For example, `assume(a>0);` tells Maxima that `a` is positive.

Maxima has no simple symbol for 'not equal to', so you have to use the `notequal` command for this, as above.

(ii) Now integrate x^n using

```
integrate(x^n, x);
```

No question is asked this time, since Maxima knows all it needs to know about `n`.

(iii) Enter

```
facts();
```

to see all the additional facts about variables known to Maxima.

This lists all such facts. The only one shown should be the one you gave earlier.

(iv) Tell Maxima to forget about the assumption on `n` by entering

```
forget(notequal(n,-1));
```

(v) Type `facts();` again to see all the facts now known.

No facts are displayed, since Maxima has now forgotten the assumption about n.

The commands used in the previous activity are summarised below, together with the `equal` command, which is used in a similar fashion to `notequal`.

Facts about variables

Operation	Command	Example
Make an assumption about a variable	`assume( )`	`assume(a>0);`
State two things are equal	`equal( , )`	`assume(equal(n,3));`
State two things are not equal	`notequal( , )`	`assume(notequal(n,-1));`
Forget a property	`forget(property)`	`forget(a>0);`
List all known facts	`facts()`	`facts();`
List all known facts about a particular variable	`facts(variable)`	`facts(a);`

Quadrature

You saw in Computer activity 28(g) that Maxima cannot always find an antiderivative for a function. However, if Maxima cannot find an antiderivative of a function f, it can often find an *approximate* value for a definite integral of the form $\int_a^b f(x)\,dx$ by calculating a sophisticated numerical approximation to the area between the graph of the function and the x-axis. The command for doing this is `quad_qags`, which is demonstrated in the following activity.

The 'quad' part of the name of this command arises from the fact the process of finding the value of a definite integral (or any area) is sometimes called **quadrature**. The letters `q`, `a`, `g` and `s` in the second half of the name specify the particular method Maxima uses to find the approximate value of the definite integral. This method gives good results for a wide variety of integrals.

Computer activity 31 Finding approximate values of definite integrals

(a) Try to calculate $\int_0^1 \ln(\sin(x^3))\,dx$ by entering

```
integrate(log(sin(x^3)), x, 0, 1);
```

The integral is returned unevaluated, since Maxima cannot calculate it.

(b) Find an approximate value of the integral by entering

```
quad_qags(log(sin(x^3)), x, 0, 1);
```

The `quad_qags` command calculates an approximate value for a definite integral.

Notice that the output of the `quad_qags` command is a list. The first element of the list is the approximate value of the definite integral, the second element is an estimate of the accuracy of the approximation, the third element is the number of values at which the function to be integrated was evaluated during the calculation, and the final element is an error code. An error code of 0 means that no errors occurred.

Finding approximate values for definite integrals

Operation	Command	Example
Find an approximate value of a definite integral	`quad_qags(▢,variable,lower limit,upper limit)`	`quad_qags(exp(-x^2),x,0,1);`

If you began studying this section from Activity 46 of Unit 1, this completes your study of Unit 1.

4 Number theory

Many of the calculations that you have seen in Unit 3 can be performed quickly and accurately by a computer, which is particularly useful when the numbers are large. Maxima is ideally suited to this, since it can do exact calculations with extremely large numbers. However, for really large numbers some calculations may take a long time!

The Maxima commands for finding the quotient and remainder when one integer is divided by another, and the highest common factor (HCF) of two integers, are demonstrated in the following activity.

Computer activity 32 Finding quotients, remainders and HCFs

(a) Find the quotient on dividing 59 by 8, by entering

```
quotient(59,8);
```

(b) Find the remainder on dividing 59 by 8, by entering

```
remainder(59,8);
```

> You could also use the command `divide(▢,▢)` which gives a list containing the quotient and remainder of the given numbers.

(c) Find the quotient and remainder when 3^{100} is divided by $2^{100} + 11$.

(d) Find the highest common factor of 108 and 93, by entering

```
gcd(108,93);
```

> The name of the Maxima command for the HCF is `gcd`, because an alternative name for the highest common factor is 'greatest common divisor'.

(e) Find the highest common factor of 3^{100} and $2^{100} + 11$.

You know from Unit 3 that if a and b are integers, not both 0, and d is their highest common factor, then there exist integers v and w such that $av + bw = d$. This is known as *Bézout's identity.* Maxima has a command for finding d, v and w for a given a and b, which is demonstrated in the following activity.

Computer activity 33 Using Bézout's identity

(a) Find the highest common factor d of 108 and 93, and two integers v and w such that $108v + 93w = d$, by entering

```
gcdex(108,93);
```

The result of this calculation is a list:

```
[-6,7,3]
```

The last number in the list is the highest common factor of the numbers, which agrees with the result of Computer activity 32(d). The first two numbers are the required integers v and w, so

$$108 \times (-6) + 93 \times 7 = 3.$$

(b) Find the highest common factor d of 3^{100} and $2^{100} + 11$, and two integers v and w such that $3^{100}v + (2^{100} + 11)w = d$.

The commands introduced in this section are summarised below.

Quotients, remainders and highest common factors

Operation	Syntax	Example
Quotient on dividing one number by another	`quotient(▨,▨)`	`quotient(6,4);`
Remainder on dividing one number by another	`remainder(▨,▨)`	`remainder(6,4);`
Quotient and remainder on dividing one number by another	`divide(▨,▨)`	`divide(6,4);`
Highest common factor (greatest common divisor) of two numbers	`gcd(▨,▨)`	`gcd(6,4);`
Bézout's identity: given integers a and b find the highest common factor d and integers v and w such that $av + bw = d$	`gcdex(▨,▨)`	`gcdex(6,4);`

If you began studying this section from Activity 6 of Unit 3, return to the unit to continue your study.

5 Parametric equations

In this section you will learn how to use Maxima to plot curves given by parametric equations.

5.1 Plotting curves from parametric equations

Subsection 3.2 of this *Guide* reviewed how to use the `wxplot2d` command to plot graphs of equations given in the explicit form $y = f(x)$, and how to use the `wximplicit_plot` command to plot graphs of equations given in implicit form. This subsection reviews how to use the `wxplot2d` command to plot curves given by parametric equations.

Note that if you are using a Maxima interface other than wxMaxima, you should use the `plot2d` command instead of `wxplot2d`. The two commands are used in similar ways.

Computer activity 34 Plotting a curve from parametric equations

(a) Plot the curve given by the parametric equations

$$x = t + 4, \quad y = 2t + 3$$

for values of the parameter t in the range $-1 \leq t \leq 2$ by entering

```
wxplot2d( [parametric, t+4, 2*t+3, [t,-1,2]] );
```

Here, the first argument of the `wxplot2d` command is a list whose first element is the keyword `parametric`. The second and third elements of the list are the parametric expressions for x and y, respectively. The fourth element of the list is itself a list containing the name of the parameter followed by the minimum and maximum values of the parameter for which the curve should be plotted. It is important that this list is *within* the list beginning `parametric`.

The curve is a line segment corresponding to the specified range of values of t, namely $-1 \leq t \leq 2$. (It is part of the straight line that you plotted in Activity 23 of Unit 4.) Maxima automatically chooses ranges for the x- and y-axes to ensure that the whole of the line segment is visible.

(b) To see more clearly that the curve plotted in part (a) is a line segment rather than a whole line, plot the line again, this time with the x- and y-axes having the ranges $0 \leq x \leq 8$ and $0 \leq y \leq 8$, by entering

```
wxplot2d( [parametric, t+4, 2*t+3, [t,-1,2]],
[x,0,8], [y,0,8], [xlabel,"x"], [ylabel,"y"] );
```

Here, the usual Maxima graph plotting options have been used to specify the ranges of the x- and y-axes. The graph now clearly shows the line segment described by the given equations and range of t.

The command also used the usual plotting options to set the labels for the x- and y-axes, which replace the expressions for the x- and y-coordinates of the points on the line which were displayed on the graph produced in part (a).

(c) Plot the curve given by the parametric equations

$$x = 3t^2, \quad y = 6t \quad (-2 \leq t \leq 2).$$

This is the curve that you plotted by hand in Activity 24 of Unit 4.

When you are plotting curves from parametric equations, it is often convenient to assign the expressions for x and y to variables or functions, and use these in the `wxplot2d` command. Note, however, that you should *not* assign them to the *variables* `x` and `y`, as this will give errors when you specify ranges for the axes. You can, however, assign them to the *functions* x and y, as demonstrated next.

Computer activity 35 Using functions to plot curves from parametric equations

Plot the curve given by

$$x = t\cos t, \quad y = t\sin t \quad (0 \leq t \leq \pi)$$

as follows.

(a) Define the function `x(t)` to be $t\cos t$ and the function `y(t)` to be $t\sin t$.

(b) Plot the curve by entering

```
wxplot2d( [parametric, x(t), y(t), [t,0,%pi]],
[xlabel,"x"], [ylabel,"y"] );
```

Computer activity 36 Plotting more curves from parametric equations

Plot the following curves given using parametric equations.

(a) $x = 12 - 12\sin^2 t, \quad y = 12\cos t \quad (0 \leq t \leq \pi)$

(b) $x = t\sin(2t), \quad y = t\cos(2t) \quad (0 \leq t \leq 10)$

(c) $x = \sin(2t), \quad y = \cos(3t) \quad (0 \leq t \leq 20)$

This curve is an example of a *Lissajous curve*. In general, a Lissajous curve is a curve given by parametric equations of the form

$$x = A\sin(at + d), \quad y = B\sin(bt)$$

where A, B, a, b and d are numbers. Different values of these numbers give different curves. If a/b is a rational number, then a closed curve is obtained, provided it is plotted for a sufficiently large range of t.

You can plot a curve from parametric equations on the same diagram as another curve from parametric equations, or on the same diagram as any other graph. To do this, list the curves to be plotted as the first argument of the `wxplot2d` command. In this case, to prevent the commands getting too long, it is often helpful to assign the list defining a parametric curve to a named variable, as demonstrated in the following activity.

Computer activity 37 Plotting more than one curve

(a) Plot the curve given by

$$x = 3t^3, \quad y = 8t^2 \quad (-1 \leq t \leq 1)$$

and the curve that is the graph of the function

$$f(x) = 2x^2$$

on the same diagram, as follows.

(i) First, assign the expressions for x and y for the first curve to the functions `x(t)` and `y(t)`, and define `f(x)` to be $2x^2$.

(ii) Assign the list required to specify the first curve to the variable `p` by entering

```
p:[parametric, x(t), y(t), [t,-1,1]];
```

(iii) Plot the curves by entering

```
wxplot2d( [p,f(x)], [x,-2,2]);
```

Here, the Maxima specifications of the two curves are given in a list as the first argument of `wxplot2d`. This is a similar syntax to that used when plotting the graphs of more than one function together.

Note that since one of the plots is specified by a function, a range of values of x over which to plot is needed, as this is a requirement of `wxplot2d` when plotting functions. This must be given immediately after the list of curves to plot, as the second argument of the `wxplot2d` command.

(b) Plot on the same diagram the two curves given by

$$x = \sin^3 t, \quad y = \cos^3 t \quad (0 \leq t \leq 2\pi)$$

and

$$x = \cos^2 t \sin t, \quad y = \sin^2 t \cos t \quad (0 \leq t \leq 2\pi)$$

The use of the `wxplot2d` command to plot curves from parametric equations is summarised below.

Plotting curves from parametric equations

Operation	Command	Example
Plot a curve from parametric equations	`wxplot2d([parametric, expression for x , expression for y , [parameter , min , max ]], ...)`	`wxplot2d([parametric, 2*t, 3*t+1, [t,0,1]]);`

Note: the parameter can be any variable, but `x` and `y` should not be used.

If you began studying this subsection from Activity 25 of Unit 4, return to the unit to continue your study.

5.2 Plotting conics parametrically

You can plot conics parametrically using the `wxplot2d` command as described in Subsection 5.1 of this *Guide*.

However, you need to take care when plotting a hyperbola, since the expressions for x and y in the standard parametrisation of a hyperbola in standard position which is of the form

$$x = a \sec t, \quad y = b \tan t \quad \left(-\frac{\pi}{2} < t < \frac{\pi}{2}, \frac{\pi}{2} < t < \frac{3\pi}{2} \right)$$

are not defined at the endpoints $t = -\pi/2$, $t = \pi/2$ and $t = 3\pi/2$ of the ranges of values of t. If you try to use Maxima to plot a hyperbola using a parametrisation of this form, then the system will attempt to evaluate the expressions at values of t near $-\pi/2$, $\pi/2$ and $3\pi/2$, and will obtain values with very large magnitudes, resulting in a misleading graph. You can correct this by imposing suitable ranges for the x- and y-axes, as demonstrated in the next activity.

Another advantage of imposing suitable ranges for the x- and y-axes is that, as demonstrated in the next activity, you do not have to plot *two* curves, one for $-\pi/2 < t < \pi/2$ and one for $\pi/2 < t < 3\pi/2$. Instead you can plot a single curve for $-\pi/2 < t < 3\pi/2$. This is because the imposed ranges for the x- and y-axes prevent Maxima displaying points whose coordinates have large magnitudes arising from evaluating the expressions at values of t near $\pi/2$.

Computer activity 38 Plotting conics parametrically

(a) Plot the hyperbola given by

$$x = 3 + 2\sec t, \quad y = -5 + \tan t \quad \left(-\frac{\pi}{2} < t < \frac{\pi}{2}, \frac{\pi}{2} < t < \frac{3\pi}{2}\right)$$

by entering

```
wxplot2d( [parametric, 3+2*sec(t), -5+tan(t),
[t,-%pi/2,3*%pi/2]], [xlabel,"x"], [ylabel,"y"] );
```

This is the hyperbola considered in Activity 40(a) of Unit 4.

The graph is distorted by the large values calculated near $t = -\frac{\pi}{2}$, $t = \frac{\pi}{2}$ and $t = \frac{3\pi}{2}$.

(b) Plot the hyperbola in part (a) again, this time restricting the x- and y-axes to $-2 \leq x \leq 8$ and $-10 \leq y \leq 0$.

(c) Plot the hyperbola in parts (a) and (b) a third time, now including the asymptotes, which are $y = \frac{x}{2} - \frac{13}{2}$ and $y = -\frac{x}{2} - \frac{7}{2}$, on the diagram.

(d) Plot the parabola given by

$$x = -1 + 4t^2, \quad y = 4 + 8t \quad (-2 \leq t \leq 2)$$

This is the parabola considered in Activity 40(b) of Unit 4.

(e) Plot the ellipse given by

$$x = 1 + 2\cos t, \quad y = 2 + 3\sin t \quad (0 \leq t \leq 2\pi)$$

If you began studying this subsection from Activity 41 of Unit 4, this completes your study of Unit 4.

6 Partial fractions and polynomial division

In this section you will learn how to use Maxima to find partial fraction expansions of rational expressions, and to find the quotient and remainder when you divide one polynomial expression by another polynomial expression.

6.1 Partial fractions

The Maxima command for finding the partial fraction expansion of a rational expression is `partfrac`. This command is demonstrated in the following activity.

Computer activity 39 Finding a partial fraction expansion

(a) Find the partial fraction expansion of the rational expression $\dfrac{7x}{(x+5)(x-2)}$ by entering

```
partfrac( 7*x/((x+5)*(x-2)), x );
```

The first argument of the `partfrac` command is the rational expression that you want to write in terms of its partial fractions. The second argument is the variable used in the rational expression. All the rational expressions we consider here contain only one variable.

Notice that in this example the denominator of the rational expression is enclosed in brackets, since without the brackets `7*x/(x+5)*(x-2)` would be interpreted as

$$\frac{7x(x-2)}{x+5}.$$

(b) Confirm that the partial fraction expansion is equal to the original expression by entering

```
factor(%);
```

This command adds together the partial fractions obtained in part (a) and expresses the result with a factorised denominator. Remember that `%` refers to the result of the last evaluated command.

The output is the rational expression that you started with.

The use of the `partfrac` command is summarised below.

Partial fraction expansion of a rational expression

Operation	Command	Example
Find the partial fraction expansion of a rational expression	`partfrac(expression, variable)`	`partfrac(1/(x^2-1),x);`

In the following activity you can use the `partfrac` command to see the form of the partial fraction expansions of various rational expressions.

Computer activity 40 Investigating partial fraction expansions

Find the partial fraction expansions of the following rational expressions.

How are the denominators of the partial fractions related to the denominators of the original expressions?

(a) $\dfrac{9}{(2x+1)(x-1)}$ (b) $\dfrac{3x}{x^2+x-2}$ (c) $\dfrac{4x+8}{(4x-3)^2}$

(d) $\dfrac{8x+4}{(5x-3)(x-1)^2}$ (e) $\dfrac{26}{(x+5)(x^2+1)}$

> Notice that all the partial fractions that you obtained in parts (a)–(e) are of the form
>
> $$\frac{A}{(ax+b)^n} \quad \text{or} \quad \frac{Ax+B}{(ax^2+bx+c)^n}$$
>
> where n is a positive integer and a, b, c, A and B are constants, as described in Unit 7.

If you began studying this subsection from Activity 8 of Unit 7, return to the unit to continue your study.

6.2 Polynomial division

You can find the quotient and remainder on dividing one polynomial expression by another using the same commands that you use to find the quotient and remainder when dividing integers, which you met in Section 4 of this *Guide*. The use of these commands for polynomial expressions is summarised below and is demonstrated in the activity that follows.

Polynomial division

Operation	Command	Example
Quotient on dividing one polynomial expression by another	`quotient(□,□)`	`quotient(x^2,x-1);`
Remainder on dividing one polynomial expression by another	`remainder(□,□)`	`remainder(x^2,x-1);`
Quotient and remainder on dividing one polynomial expression by another	`divide(□,□)`	`divide(x^2,x-1);`

Computer activity 41 Dividing polynomial expressions

(a) Find the quotient and remainder on dividing x^5+x+1 by x^3+2, by entering

```
divide(x^5+x+1, x^3+2);
```

(b) Find the quotient and remainder on dividing $a(x)$ by $b(x)$ for each of the following pairs of polynomial expressions.

(i) $a(x) = 3x^4 - 7x, \quad b(x) = x - 1$

(ii) $a(x) = x^6 + x^5, \quad b(x) = 2x^2 + 1$

If you began studying this subsection from Activity 16 of Unit 7, return to the unit to continue your study.

7 Differential equations

In this section you will learn how to use Maxima to find analytic solutions of first-order differential equations (where this is possible), plot direction fields of first-order differential equations, and calculate approximate numerical solutions of first-order differential equations.

7.1 Solving differential equations analytically

The Maxima command for solving a differential equation analytically is `ode2`. Here, '`ode`' stands for *ordinary differential equation*. All the differential equations that you have met in Unit 8 are so-called *ordinary* differential equations, which simply means that the unknown function is a function of only one variable. Equations in which the unknown function is a function of more than one variable are called *partial differential equations* and you may meet these in higher-level modules. The '`2`' in the command name means that this command is capable of analytically solving (where possible) differential equations of order up to 2; that is, it can solve first- and second-order differential equations. However, as in Unit 8, we will consider only first-order differential equations here.

When you enter a differential equation in Maxima, you need to use a special syntax to enter the derivative in the equation. Consider, for example, the differential equation

$$\frac{\mathrm{d}y}{\mathrm{d}x} - y = e^x \sin x,$$

which you met in Example 11 of Unit 8. You might think that you can enter the term $\mathrm{d}y/\mathrm{d}x$ by typing `diff(y,x)`, since the command for differentiating an expression in Maxima is `diff`. (This is revised in Subsection 3.4 of this *Guide*.) However, entering `diff(y,x)` in Maxima gives the result `0`. You might like to try this. The reason is that when you enter `diff(y,x)`, Maxima will try to perform the differentiation, and since there is nothing to tell it that `y` is a function of `x`, it will obtain the result `0`.

To enter the operation of differentiation in Maxima, while preventing Maxima from trying to actually perform the operation, you need to prefix the `diff` command with a single quote mark (`'`). You may recall (again, as revised in Subsection 3.4 of this *Guide*) that two quote marks are used to *force* the evaluation of a command. Here, one quote mark is used to *prevent* the evaluation of a command.

The use of this method of entering differential equations and the `ode2` command are demonstrated in the following activity.

Computer activity 42 Solving differential equations analytically

(a) Solve the differential equation

$$\frac{\mathrm{d}y}{\mathrm{d}x} - y = e^x \sin x$$

as follows.

(i) First assign the equation to the variable `eqn` by entering

```
eqn:'diff(y,x)-y=exp(x)*sin(x);
```

Notice the use of `'diff(y,x)` for the derivative $\mathrm{d}y/\mathrm{d}x$, to prevent Maxima from performing the differentiation immediately.

The quote mark to use is the upright one (`'`) on your keyboard, not the sloping one (`` ` ``).

wxMaxima displays `'diff(y,x)` as $\frac{\mathtt{d}}{\mathtt{d\,x}}\mathtt{y}$ when showing the result of this command. Also, the expression `exp(x)` is displayed as `%e`$^{\mathtt{x}}$, where `%e` is Maxima's notation for the constant e.

(ii) Now solve the equation by entering

```
ode2(eqn, y, x);
```

The first argument of the `ode2` command is the equation to be solved, or, as in this case, the name of a variable representing that equation. The second argument is the name of the dependent variable, and the third argument is the name of the independent variable.

The result of this command is the general solution of the differential equation. The symbol `%c` is used to represent the arbitrary constant. Notice that the solution is an *equation* relating the dependent variable, `y`, to the independent variable, `x`.

The solution should be the same as that obtained in Example 11 of Unit 8, namely $y = e^x(-\cos x + c)$.

(b) Use Maxima to solve the differential equation

$$3\frac{\mathrm{d}y}{\mathrm{d}x} + y = xy^2$$

You might like to use the `fullratsimp` command to simplify the result.

(c) Use Maxima to try to solve the differential equation

$$\frac{\mathrm{d}y}{\mathrm{d}x} = e^{\cos x} - 1$$

This differential equation is directly integrable, but Maxima cannot calculate one of the integrals needed for the solution. The answer is displayed with this integral unevaluated.

(d) Use Maxima to try to solve the differential equation

$$\frac{\mathrm{d}y}{\mathrm{d}t} = ty^3 - 1$$

Notice that the independent variable is t, not x.

The result of the `ode2` command is `false`, which means that Maxima is unable to solve the equation using this command.

The `ode2` command uses a range of methods to try to find an analytic solution of a differential equation. These include the methods that you have studied in Unit 8 and some others. However, the command cannot always find an analytic solution to an equation, as you saw in Computer activity 42(c) and (d). One reason for this is that not all differential equations have analytic solutions. In Subsections 7.2 and 7.3 of this *Guide* you will see some ways in which you can investigate solutions if analytic solutions cannot be found.

As you have seen in Unit 8, having found the general solution to a differential equation, you often need to find a particular solution that satisfies a given condition. The command `ic1` can do this, as demonstrated in the following activity. The '`ic`' in the name of this command stands for 'initial condition', which is how we often describe the additional information needed to find a particular solution. The number '`1`' means that this command is appropriate for first-order differential equations, where one initial condition is needed to allow you to find a value for the one arbitrary constant appearing in the general solution.

Computer activity 43 Solving an initial value problem

Find the solution of the initial value problem

$$t\frac{\mathrm{d}y}{\mathrm{d}t} + 2y = t^2, \qquad \text{where } y(1) = 1,$$

as follows.

(a) Assign the equation to the variable `eqn`.

(b) Solve the equation, and assign the solution to the variable `sol`, by entering

```
sol:ode2(eqn, y, t);
```

(c) Find the required particular solution of the equation by entering

```
ic1(sol, y=1, t=1);
```

> The given initial condition, $y(1) = 1$, is equivalent to saying that $y = 1$ when $t = 1$.
>
> The first argument of the `ic1` command is the general solution previously calculated. The second and third arguments give the initial condition by specifying corresponding values of the dependent and independent variables, in either order. These values are given by equations of the form
>
> ```
> variable name = value
> ```
>
> The solution obtained should be the same as the one you found by hand in Activity 27(b) of Unit 8, namely $y = t^2/4 + 3t^{-2}/4$.

The commands needed to solve differential equations analytically are summarised below.

Solving differential equations analytically

Operation	Command	Example
Prevent the evaluation of a command	`'`	`'diff(y,x)`
Solve a differential equation analytically	`ode2( equation , dependent variable , independent variable )`	`ode2('diff(y,x)=x, y, x);`
Apply initial conditions to the analytic solution of a differential equation	`ic1( solution , variable = value , variable = value )`	`ic1(sol, y=1, x=0);`

Here is another problem for you to solve using Maxima.

Computer activity 44 Solving another initial value problem

(a) Solve the initial value problem

$$\frac{dy}{dx} = y - (x-2)^2, \qquad \text{where } y(0) = 1.9.$$

(b) Plot a graph of the solution over the range $0 \leq x \leq 5$.

(c) Find the exact value of y given by the solution when $x = 4$.

7.2 Plotting direction fields

In Unit 8 you saw that one way of investigating the solutions of a differential equation when it is not possible to find analytic solutions is to plot a *direction field*, as illustrated in Figure 11.

A direction field of a differential equation of the form

$$\frac{\mathrm{d}y}{\mathrm{d}x} = f(x, y),$$

where $f(x, y)$ is a known function, is a set of short line segments equally spaced in the x, y-plane. The gradient of each segment is equal to the value of $f(x, y)$ at the midpoint of that segment. The direction field gives you an idea of the shape of the graph of any particular solution of the differential equation. This is because at each point (x, y) on the graph of a particular solution the gradient is $f(x, y)$. So where such a graph passes through the midpoint of a line segment, the graph will have the same gradient as the line segment, and where the graph passes close to a line segment, you can expect the graph to have nearly the same gradient as the line segment.

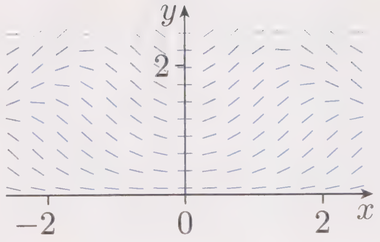


Figure 11 A direction field for $\mathrm{d}y/\mathrm{d}x = \sin(xy)$

The Maxima command for plotting a direction field is `wxdrawdf`, where '`df`' stands for 'direction field'. This command is not part of the basic Maxima collection of commands, so before using it you have to load an additional Maxima package, the `drawdf` package. This package needs to be loaded only once in each Maxima session. Details of how to load the package are given in Computer activity 45.

If you are using a Maxima interface other than wxMaxima, then you should use the `drawdf` command instead of `wxdrawdf`. If you have configured Maxima to automatically use different colours and styles for graphs, as described in the Accessibility section of the OU Maxima website, then you should use the commands `ouwxdrawdf` (if you use wxMaxima) or `oudrawdf` (otherwise) instead. All these commands are used in a similar way to `wxdrawdf`, and all of them require the `drawdf` package to be loaded.

Note that the `wxdrawdf` command uses a different drawing system to the `wxplot2d` and `wximplicit_plot` commands that you have used previously, so many of the options that you have used with `wxplot2d` and `wximplicit_plot` do not work with `wxdrawdf`.

The use of the `wxdrawdf` command is demonstrated in the following activity, where you will plot the direction field of the differential equation that you solved analytically in Computer activity 44.

Computer activity 45 Investigating a direction field

(a) First load the `drawdf` package by typing

```
load(drawdf);
```

Maxima will produce some output as a result of this command, which may vary depending on whether you have ever used this package on your computer before.

The output will end with a line similar to

```
(%o1) C:\maxima-5.38.1\share\maxima\5.38.1_5\share\
       diffequations\drawdf.mac
```

which indicates the location of the package on your computer.

(b) Plot the direction field of the differential equation

$$\frac{\mathrm{d}y}{\mathrm{d}x} = y - (x-2)^2,$$

by entering

```
wxdrawdf(y-(x-2)^2, [x,y], field_arrows=false);
```

The first argument of the wxdrawdf command is the expression for the gradient $\frac{\mathrm{d}y}{\mathrm{d}x}$ at each point (x, y).

The second argument is a list containing the names of the two variables in the differential equation. In this case these are x and y. The first variable listed is plotted along the horizontal axis and the second is plotted along the vertical axis. Note that if, as in this case, the variables are x and y and you want to plot x on the horizontal axis and y on the vertical axis, then this argument can be omitted.

The third argument, field_arrows=false, indicates that you want to plot a set of line segments not arrows. As you saw in Unit 8, a direction field is drawn using line segments. In other applications, arrows are needed.

Notice that the default ranges of the values of the variables plotted by the wxdrawdf command are from -10 to 10 for both variables.

From the direction field displayed you can see that for much of the region plotted the line segments are nearly vertical. Something more interesting seems to be happening in one particular region of the direction field however, so in the next part of this activity you can change the region plotted to concentrate on this area.

(c) Plot the same direction field, this time limiting the range of values of x and y to $-3 \leq x \leq 6$ and $-5 \leq y \leq 10$, respectively, by entering

```
wxdrawdf(y-(x-2)^2, [x,-3,6], [y,-5,10],
 field_arrows=false);
```

(You may like to edit and re-evaluate the command that you entered in part (b) rather than entering the command again.)

The ranges of values of x and y are specified by including additional arguments each of which is a list containing the name of the variable and the minimum and maximum values of this variable over which the direction field should be plotted. This way of specifying ranges is identical to the way that you have previously specified ranges of values when using the `wxplot2d` and `wximplicit_plot` commands.

Note that when you specify ranges of values for both variables, you can omit the argument listing the variables, even if they are not x and y.

In the plot, you can more easily see details of the behaviour of solutions of the differential equation in the region shown.

(d) Use the direction field to try to work out what the graphs of particular solutions of the differential equation might look like.

To do this, start at a particular point on the direction field and move in the direction given by the line segment at that point. When you get to the next column of line segments, continue moving but in the direction suggested by the nearest line segments. Continue in this way, ideally 'smoothing' the changes of direction, until you reach an edge of the plot.

You may like to print the direction field and draw such graphs of solutions on it.

(i) What do you think the graph of the particular solution that passes through the point $(0, 0)$ might look like?

(ii) What do you think the graph of the particular solution that passes through the point $(0, 4)$ might look like?

(e) Add the following additional argument to the command entered in part (c):

```
[trajectory_at, 0,0]
```

This argument adds the approximate graph of the particular solution that passes through the point specified.

The argument is a list beginning with the keyword `trajectory_at`. (Note the underscore character in this keyword.) The second and third elements of the list are the coordinates of the point that the required solution should pass through.

You can add more than one approximate solution graph to the direction field by including more than one argument of this form.

Each approximate particular solution is made up of a sequence of small straight line segments, which are found as follows. Maxima starts with the specified point. It then increases the value of the independent variable (in this case x) by a small amount, known as the *step size*, and calculates the corresponding change in the dependent variable (in this case y), by using the value of $\mathrm{d}y/\mathrm{d}x$ at that point. This gives a new point. The process is then repeated starting with this new point, and repeated again a number of times, to give a sequence of points. Maxima then returns to the specified point and carries out the same process, this time *decreasing* the independent variable by the step size. The points obtained are joined up with short line segments.

Maxima automatically changes the step size used when calculating approximate solutions depending on how rapidly the direction field is changing. You can set the maximum step size to be used by including an additional argument of the form

```
[tstep, 0.001]
```

This example sets the maximum step size to be used to 0.001.

You can also change the number of steps taken by including an additional argument of the form

```
[nsteps, 200]
```

which sets the number of steps to be taken to 200.

Note that these additional arguments only affect `trajectory_at` arguments that appear *after* them in the list of arguments of the `wxdrawdf` command.

You can see that the particular solution that passes through the point $(0, 0)$ has y decreasing as x increases.

(f) Add to the plot the particular solution that passes through the point $(0, 4)$.

Notice that this particular solution has a local minimum near $x = 0$.

(g) Finally, add to the plot the particular solution that passes through the point $(0, 1.9)$.

This graph should match the graph of the particular solution that you obtained analytically and plotted in Computer activity 44.

The solutions plotted here demonstrate that the behaviour of the solution to a differential equation can vary depending on the initial conditions.

The fact that the behaviour of the solutions of a differential equation can be obtained from a direction field without the need to solve the differential equation makes a direction field a powerful tool.

The commands needed to plot direction fields are summarised in the tables below.

Plotting direction fields

Operation	Command	Example
Plot the direction field of a differential equation	`wxdrawdf(expression, [variable, variable], field_arrows=false, ...)`	`wxdrawdf(x*y,[x,y], field_arrows=false);`

Note 1: the `drawdf` package needs to be loaded before this command can be used.

Note 2: the first variable listed is plotted on the horizontal axis and the second variable listed is plotted on the vertical axis.

Direction field plotting options

Operation	Argument	Example
Set the range of a variable	`[variable, ▨, ▨]`	`[x,0,5]`
Plot the graph of a particular solution	`[trajectory_at, ▨, ▨]`	`[trajectory_at,0,1]`
Set the maximum step size to use when approximating a solution	`[tstep, ▨]`	`[tstep,0.001]`
Set the number of steps used when approximating a solution	`[nsteps, ▨]`	`[nsteps,200]`

In the following activity you can use direction fields to help understand the behaviour of the solutions of two differential equations that Maxima is unable to solve analytically.

Computer activity 46 Using direction fields

Plot a direction field for each of the following differential equations and then add the graphs of some approximate particular solutions and describe their behaviour. For example, you might like to start by considering the solution that passes through $(0, 0)$.

How does the behaviour of the particular solutions vary as the initial condition is changed?

(a) $$\frac{dy}{dx} = e^{\cos x} - 1$$

(Hint: use the default range of the direction field, $-10 \leq x \leq 10$, $-10 \leq y \leq 10$.)

(This is the differential equation that Maxima was unable to solve analytically in Computer activity 42(c).)

(b) $$\frac{dy}{dt} = ty^3 - 1$$

(Hint: restrict the range of the direction field to $-2 \leq t \leq 2$, $-2 \leq y \leq 2$.)

(This is the differential equation that Maxima was unable to solve analytically in Computer activity 42(d).)

7.3 Solving differential equations numerically

Carl Runge (1856–1927)

Martin Wilhelm Kutta (1867–1944)

As you saw in the previous section, you can find an approximate particular solution of a differential equation by using the direction field of the equation.

Such a solution is usually represented by a sequence of points that lie on the approximate solution. Such approximate solutions are known as **numerical solutions**.

The Maxima command for numerically solving a differential equation of the form

$$\frac{dy}{dx} = f(x, y)$$

is `rk`.

Here, '`rk`' stands for *Runge–Kutta*. This is the name given to an often-used class of methods for solving differential equations numerically. They are named after the German mathematicians Carl Runge and Martin Wilhelm Kutta, who first developed them.

The `rk` command is demonstrated in the following activity.

To solve a differential equation numerically, we need both an equation and an initial condition, since the approximate solution is an approximation of a *particular* solution of the equation. That is, we numerically solve an initial value problem.

Computer activity 47 Solving an initial value problem numerically

(a) Find a numerical approximation of the solution of the initial value problem

$$\frac{dy}{dx} = y - (x-2)^2, \qquad \text{where } y(0) = 1.9,$$

for $0 \leq x \leq 5$ by entering

```
pts:rk(y-(x-2)^2, y, 1.9, [x, 0, 5, 1]);
```

(This is the differential equation that you solved analytically in Computer activity 44.)

The first argument of the `rk` command is the expression for $\frac{dy}{dx}$ given by the equation to be solved.

The second argument is the dependent variable (in this case y) and the third argument is the initial value of this variable, as given by the initial condition.

The final argument is a list beginning with the name of the independent variable (in this case x) followed by the initial value of this variable, as given by the initial condition, then the last value of the variable for which you want to calculate the solution, and finally the step size to be used. The step size is the change in the independent variable between adjacent points in the numerical solution.

The result is a list of pairs of numbers, in which each pair consists of the x- and y-coordinates of a point on the numerical solution.

Notice that the x-coordinates start at 0 and increase in steps of 1 until 5, as requested in the command entered.

(b) The analytic solution of the initial value problem considered in part (a) that was obtained in Computer activity 44 is

$$y = -\frac{e^x}{10} + x^2 - 2x + 2.$$

Assign the analytic solution to the variable `asol`, and then plot this solution together with the numerical solution calculated above by entering

```
asol:-exp(x)/10+x^2-2*x+2;
```

then

```
wxplot2d([asol, [discrete, pts]], [x,0,5],
[legend, "Analytic", "Numerical"]);
```

Here, the first argument of the `wxplot2d` command is a list of the curves to plot. (This use of the command is revised in Computer activity 18 in Subsection 3.2 of this *Guide*.) The first element of this list is the analytic solution and the second element is a list that represents the numerical solution. This list consists of the keyword `discrete` followed by the list of points to be plotted. In the plot, these points will be joined by straight line segments. A legend has also been included.

Notice that the numerical solution does indeed approximate the analytic solution, but the agreement gets worse as x increases. This is not surprising since only a crude numerical solution was calculated, using the relatively large step size of 1.

(c) Calculate a revised numerical solution, using a step size of 0.1, and plot this solution together with the analytic solution.

> The agreement between the two solutions is much improved.
>
> Usually, the smaller the step size in a numerical solution, the better the result.

The `rk` command for finding a numerical solution of a differential equation and the method used to plot the graph represented by a set of points are summarised below.

Solving a differential equation numerically

Operation	Command	Example
Solve a differential equation numerically	`rk(expression, dependent variable, initial value, [independent variable, initial value, final value, step size])`	`rk(x^2,y,1, [x,0,10,0.1]);`

Plotting graphs represented by a set of points

Operation	Command	Example
Plot a graph represented by a set of points	`wxplot2d([discrete, list of points])`	`wxplot2d([discrete, [[0,0], [1,1]] ]);`

Here is an activity for you to try.

Computer activity 48 Finding more numerical solutions

Find a numerical solution of each of the following initial value problems, over the range given. Plot your solutions and compare them to the direction fields plotted in Computer activity 46.

(a) $\dfrac{\mathrm{d}y}{\mathrm{d}x} = e^{\cos x} - 1$ where $y(0) = 0$, for $0 \leq x \leq 10$

(b) $\dfrac{\mathrm{d}y}{\mathrm{d}t} = ty^3 - 1$ where $y(0) = 1.5$, for $0 \leq t \leq 1$

If you began studying this section from Activity 37 of Unit 8, this completes your study of Unit 8.

8 Eigenvalues and eigenvectors

In this section you will learn how to use Maxima to find the characteristic equation, eigenvalues and eigenvectors of a matrix. Before doing that, in Subsection 8.1 you can revise how to enter and use matrices in Maxima. You learned how to do this in MST124 Unit 9, and if you are still familiar with it you may wish to skip this subsection.

8.1 Matrices

There are several ways to input matrices in Maxima. Two of these are demonstrated in the following activity.

Computer activity 49 Entering matrices

(a) Assign the matrix $\begin{pmatrix} 2 & -3 & 3 \\ -2 & 1 & 2 \\ 1 & -1 & 4 \end{pmatrix}$ to the variable P by entering

```
P:matrix( [2,-3,3], [-2,1,2], [1,-1,4] );
```

The **matrix** command allows a matrix to be defined. You give each row of the matrix as a list (that is, within square brackets with the elements separated by commas).

By default, wxMaxima displays matrices using square brackets:

$$\begin{bmatrix} 2 & -3 & 3 \\ -2 & 1 & 2 \\ 1 & -1 & 4 \end{bmatrix}$$

If you are using a different interface to Maxima, or have chosen different display settings, matrices may be displayed as follows.

```
[   2  -3  3  ]
[             ]
[  -2   1  2  ]
[             ]
[   1  -1  4  ]
```

(b) If you are using wxMaxima, you can also input a matrix using an on-screen form. Use this form to assign the matrix

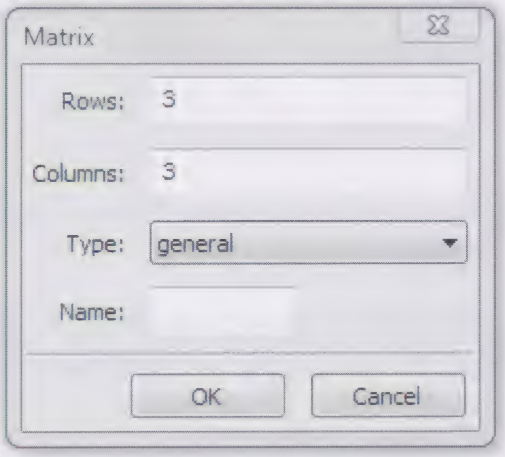
$$\begin{pmatrix} 5 & -3 \\ -1 & -2 \\ 0 & 4 \end{pmatrix}$$

to the variable Q as follows. (If you are not using wxMaxima, then enter this matrix using the **matrix** command, as in part (a).)

(i) From the `Algebra` menu select `Enter Matrix...`.

Make sure you select `Enter Matrix...`, not `Generate Matrix...`.

A window opens requesting properties of the matrix, as shown below.



Rows: 3

Columns: 3

Type: general

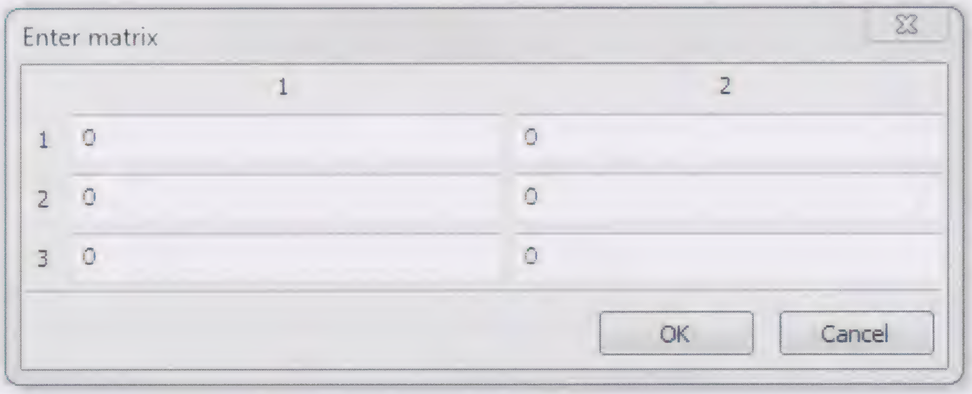
Name:

(ii) The matrix for this example has size 3 × 2, so enter 3 as the number of rows and 2 as the number of columns, replacing the default values shown.

It is also possible to give information on the type of matrix, but for now leave this set to `general`.

Finally, enter `Q` as the variable name to which the matrix will be assigned, and click 'OK'.

This opens a second window requesting the elements of the matrix.



	1	2
1	0	0
2	0	0
3	0	0

(iii) Complete entry of the matrix by filling in the elements in this second window. You may like to use the `Tab` key (→|) to move the cursor between the different elements. When you have finished, click 'OK'.

Completion of this on-screen form has the effect of including a `matrix` input line in your worksheet, followed by the appropriate output.

(c) Assign the matrix $\begin{pmatrix} 1 & -2 & -1 \\ 4 & 7 & 0 \\ -5 & 1 & 3 \end{pmatrix}$ to the variable `R` using an appropriate method.

The typed command for entering a matrix is summarised below.

Entering matrices

Operation	Command	Example
Specify a matrix	`matrix(row,row,...)`	`matrix([1,2],[3,4]);`

Note: matrices can also be entered using the `Enter Matrix...` option of the `Algebra` menu.

Once a matrix has been entered, you can use it in calculations, subject to the usual laws of matrix arithmetic. The Maxima syntax to use for matrix operations is summarised in the following table. You can practise using some of these in Computer activity 50.

Matrix operations

Operation	Syntax	Example
Matrix addition	`+`	`P+Q;`
Matrix subtraction	`-`	`P-Q;`
Scalar multiple of a matrix	`*`	`2*P;`
Matrix multiplication	`.`	`P.Q;`
Matrix powers	`^^`	`P^^3;`
Transpose (of a matrix)	`transpose(matrix)`	`transpose(P);`
Determinant (of a square matrix)	`determinant(matrix)`	`determinant(P);`

Matrix inverse (of a square matrix)	`invert(matrix)`
	or `matrix^^(-1)`
		`invert(P);`
		`P^^(-1);`

Warnings

If a matrix operation cannot be performed, for example if you attempt to add two matrices of different sizes, an error message is given.

Remember to use `.` (not `*`) for matrix multiplication and `^^` (not `^`) to find the power of a matrix.

If you use the symbol `*` when multiplying two matrices, or the symbol `^` to find a power of a matrix, then Maxima may return an answer, but in general it will be the *wrong* answer. (The symbol `*` used between two equal-sized matrices produces a matrix of the same size whose elements are the products of corresponding elements of the given matrices. The symbol ^ produces a matrix whose elements are the elements of the original matrix raised to the given power. These are sometimes called *element-wise* operations.)

Computer activity 50 Using matrix operations

In this activity,

$$\mathbf{P} = \begin{pmatrix} 2 & -3 & 3 \\ -2 & 1 & 2 \\ 1 & -1 & 4 \end{pmatrix}, \quad \mathbf{Q} = \begin{pmatrix} 5 & -3 \\ -1 & -2 \\ 0 & 4 \end{pmatrix}, \quad \text{and } \mathbf{R} = \begin{pmatrix} 1 & -2 & -1 \\ 4 & 7 & 0 \\ -5 & 1 & 3 \end{pmatrix},$$

which are the matrices you input in Computer activity 49.

(a) Find $\mathbf{P} + \mathbf{R}$.

(b) Find $3\mathbf{P}$.

This is a scalar multiple of a matrix, so use the * operator.

(c) Find $\mathbf{RQ}$.

This is matrix multiplication, so use the . operator.

(d) Find $\mathbf{P}^2$.

Since **P** is a matrix use the matrix power operator ^^.

(e) Find the determinant of **P**.

(f) Find the inverse of **R**.

(g) By multiplying **P** by $\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$, show that $\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$ is an eigenvector of **P** and find the corresponding eigenvalue.

8.2 Characteristic equations, eigenvalues and eigenvectors

You have seen in Unit 11 that the characteristic equation of an $n \times n$ matrix **A** is a polynomial equation of degree n whose solutions are the eigenvalues of **A**. For example, you saw in Example 2 of Unit 11 that the characteristic equation of the matrix $\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ is $\lambda^2 - 4\lambda + 3 = 0$.

The solutions of this equation are 1 and 3, which are the eigenvalues of the matrix.

You have also seen that the characteristic equation of an $n \times n$ matrix $\mathbf{A}$ is given by

$$\det(\mathbf{A} - \lambda \mathbf{I}) = 0,$$

where $\mathbf{I}$ is the $n \times n$ identity matrix.

The left-hand side of this equation, $\det(\mathbf{A} - \lambda \mathbf{I})$, is known as the **characteristic polynomial** of the matrix $\mathbf{A}$. For example, the characteristic polynomial of the 2×2 matrix above is

$$\det\left(\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right).$$

This can be re-written as

$$\det\begin{pmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{pmatrix}$$

or

$$\lambda^2 - 4\lambda + 3.$$

The Maxima command for finding the characteristic polynomial of a matrix is `charpoly`. It is demonstrated in the following activity. Once you know the characteristic polynomial of a matrix, it is straightforward to write down the characteristic equation.

Computer activity 51 Finding the characteristic polynomial of a matrix

(a) Find the characteristic polynomial of the matrix $\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 \\ -1 & 2 & 1 \\ -1 & -1 & 4 \end{pmatrix}$ as follows.

(i) Enter the matrix into Maxima, and assign it to the variable `A`.

(ii) Calculate the characteristic polynomial of `A` by entering

```
charpoly(A,m);
```

The first argument of the `charpoly` command is the matrix whose characteristic polynomial is wanted, and the second argument is the variable in terms of which the polynomial will be expressed.

You have seen in Unit 11 that characteristic equations are usually expressed in terms of λ, the lower-case Greek letter lambda. However, in Maxima `lambda` has a special meaning, so you need to use an alternative variable, such as `m`.

The polynomial obtained is not given in its simplest form. You can simplify it by expanding the brackets using the `expand` command.

(iii) Simplify the characteristic polynomial by entering

```
expand(%);
```

and hence write down the characteristic equation of **A**.

Remember that % refers to the output of the last evaluated command.

(b) Find the characteristic polynomial of the matrix $\mathbf{B} = \begin{pmatrix} 4 & 0 & 0 \\ 3 & 1 & -3 \\ 3 & -3 & 1 \end{pmatrix}$ and hence write down its characteristic equation.

Once you have found the characteristic equation of a matrix, you could find the eigenvalues of the matrix by solving this equation, perhaps using the `solve` command. However, Maxima also has a command **`eigenvalues`**, which you can use to calculate the eigenvalues of a matrix directly. This is demonstrated in the next activity.

First, note that the **multiplicity** of an eigenvalue of a matrix is the number of times that the eigenvalue occurs as a solution of the characteristic equation of the matrix. For example, a 2×2 matrix with characteristic equation $(\lambda - 3)^2 = 0$ has the eigenvalue 3, with multiplicity 2. Similarly, a 2×2 matrix with characteristic equation $(\lambda - 5)(\lambda - 7) = 0$ has the eigenvalue 5, of multiplicity 1, and the eigenvalue 7, of multiplicity 1. Larger matrices can have eigenvalues of greater multiplicities. For example, a 5×5 matrix with characteristic equation $(\lambda - 5)^2(\lambda - 7)^3 = 0$ has the eigenvalue 5, of multiplicity 2, and the eigenvalue 7, of multiplicity 3.

Computer activity 52 Finding the eigenvalues of a matrix

(a) Find the eigenvalues of the matrix **A** entered in Computer activity 51 by typing

```
eigenvalues(A);
```

The result of this command is a list containing two elements, each of which is itself a list.

The first of these two lists gives the (distinct) eigenvalues of the matrix, and the second list gives their respective multiplicities.

In this case, each eigenvalue has multiplicity 1.

The eigenvalues obtained should be the same as those you found in Activity 9 of Unit 11.

(b) Find the eigenvalues of the matrix **B** defined in Computer activity 51.

Internally, the `eigenvalues` command uses the `solve` command to find the solutions of the characteristic equation formed using `charpoly`. As you may recall from earlier (for example, from Subsection 3.3 of this *Guide*) the `solve` command cannot always find the solutions of an equation. In particular, the `solve` command cannot always find the solutions of a polynomial equation of degree 5 or more. This is because, in general, such equations cannot be solved using algebraic manipulations alone.

Since the characteristic equation of a $n \times n$ matrix is a polynomial equation of degree n, this means that the `eigenvalues` command cannot always find the eigenvalues of a matrix larger than 4×4. If the `eigenvalues` command cannot find the eigenvalues of a matrix, then an error message such as the following is displayed.

```
solve is unable to find the roots of
the characteristic polynomial.
```

In such circumstances, it may be possible to find numerical approximations to the eigenvalues of a matrix, but this is beyond the scope of this module.

To find the eigenvectors of a matrix, as well as the eigenvalues, you can use the `eigenvectors` command as demonstrated in the following activity.

Computer activity 53 Finding the eigenvectors of a matrix

(a) Find the eigenvectors of the matrix **A** entered in Computer activity 51 by typing

```
eigenvectors(A);
```

The result of this command is a complicated hierarchy of lists!

The first element of the main list is

```
[[1,2,3],[1,1,1]]
```

This is a list of two lists. The first of these lists gives the (distinct) eigenvalues of the matrix, and the second list gives their respective multiplicities. This is the same as the result of the `eigenvalues` command in Computer activity 52(a).

The second element of the main list is

$$[\ [[1,\frac{1}{2},\frac{1}{2}]],\ [[1,1,1]],\ [[1,1,2]]\ ]$$

(Here, additional spaces have been included to help identify the different elements of the lists.) This is a list containing three lists, one for each of the three eigenvalues. Each of these three lists contains an eigenvector corresponding to the appropriate eigenvalue.

The eigenvectors obtained should be the same as, or multiples of, the ones you verified in Activity 9 of Unit 11. Remember that any multiple of an eigenvector corresponding to an eigenvalue is also an eigenvector corresponding to that eigenvalue.

(b) Find the eigenvectors of the matrix **B** entered in Computer activity 51.

The output is similar to that for the matrix **A**, except that many of the lists contain only two elements, since **B** has only two (distinct) eigenvalues. However, there is something else to notice. Consider the second element of the main list, which is

```
[ [[0,1,1]], [[1,0,1],[0,1,-1]] ]
```

This is a list of two lists, one for each of the two eigenvalues, in order. The first of the two lists contains an eigenvector corresponding to the eigenvalue -2 (which has multiplicity 1), but the second list contains *two* eigenvectors corresponding to the eigenvalue 4, which has multiplicity 2. These two eigenvectors are not scalar multiples of each other.

In general, an eigenvalue with multiplicity 2 can have up to 2 corresponding eigenvectors that are not scalar multiples of each other.

More generally, an eigenvalue with multiplicity n can have up to n corresponding eigenvectors that together have a property called *linear independence*. You may learn about this concept in higher-level modules.

The Maxima commands for working with characteristic polynomials, eigenvalues and eigenvectors are summarised below.

Characteristic polynomials, eigenvalues and eigenvectors

Operation	Command	Example
Find the characteristic polynomial of a matrix	`charpoly(matrix,variable)`	`charpoly(A,m);`
Find the eigenvalues of a matrix	`eigenvalues(matrix)`	`eigenvalues(A);`
Find the eigenvectors of a matrix	`eigenvectors(matrix)`	`eigenvectors(A);`

If you began studying this section from Activity 10 of Unit 11, return to the unit to continue your study.

9 Solving linear recurrence systems

In this section you will learn how to use Maxima to solve constant-coefficient linear recurrence systems. First, let's revise how to input and use recurrence systems for sequences in Maxima. You first met the commands for this in MST124 Unit 10.

Computer activity 54 Working with recurrence systems and sequences

(a) Consider the first-order recurrence system

$$x_1 = 5, \quad x_n = 2x_{n-1} + 3 \quad (n = 2, 3, 4, \ldots),$$

which you met in Example 12 of Unit 12.

(i) Define this recurrence system in Maxima by entering the two commands below.

```
x[1]:5;
x[n]:=2*x[n-1]+3;
```

Notice that you assign the value of the first term, `x[1]`, using the `:` operator that is used to assign values to variables, but you specify the recurrence relation using the `:=` operator that is used to define functions.

When you enter a recurrence system in Maxima, you do not need to enter a range of values for the index variable, as Maxima assumes a suitable range. Maxima can calculate terms in a properly specified sequence for integer values of the index variable greater than or equal to the values of the index variables of the initial terms.

If you make a mistake when defining `x`, you need to remove `x` from Maxima's memory using `kill(x);` before entering the correct definition. There are more details on this in the box following this activity.

(ii) Find the second term of the sequence defined by the recurrence system above by entering

```
x[2];
```

(iii) Find the tenth term of the sequence.

(b) (i) Define the second-order recurrence system

$$u_0 = 4, \quad u_1 = 9, \quad u_n = 5u_{n-1} - 6u_{n-2} \quad (n = 2, 3, 4, \ldots)$$

in Maxima using commands similar to those used in part (a). (You met this recurrence system in Example 16 of Unit 12.)

(ii) Find the fourth and tenth terms of the sequence defined by this recurrence system.

Warning

When you are using recurrence sequences in Maxima, the system remembers all the values of the sequence that it has previously calculated, and bases subsequent calculations on them. This improves the efficiency of the calculations. However, it also means that recurrence systems cannot be changed easily.

To change the definition of a recurrence system, you first need to remove the system from Maxima's memory using the `kill` command, and then redefine it.

The commands for defining and working with recurrence sequences are summarised below.

Recurrence sequences

Operation	Command	Example

Define a recurrence sequence	`sequence[initial term number]:initial value`
	`sequence[n]:=expression containing previous terms`
		`x[1]:1;`
		`x[n]:=2*x[n-1]+x[n-2];`

Calculate a term in a recurrence sequence	`sequence[term number]`	`x[50];`

Now let's consider how to solve a recurrence system in Maxima, that is, how to find a closed form for the sequence defined by the recurrence system. The Maxima command for solving a recurrence system is `solve_rec`. This command is not part of the basic Maxima collection of commands, so before using it you have to load an additional Maxima package, the `solve_rec` package. You can do this using the `load` command. The package needs to be loaded only once in each Maxima session.

The use of the command is demonstrated in the following activity.

Computer activity 55 Solving recurrence systems

(a) First load the `solve_rec` package by entering

```
load(solve_rec);
```

Note the underscore character (_) in the name of this package.

The output of this command is a line similar to

```
(%o1) C:\maxima − 5.38.1\share\maxima\5.38.1_5\share\
      solve_rec\solve_rec.mac
```

which indicates the location of the package on your computer.

(b) Solve the recurrence system

$$x_1 = 5, \quad x_n = 2x_{n-1} + 3 \quad (n = 2, 3, 4, \ldots)$$

from Computer activity 54(a) as follows.

(i) First, remove any definition of x from the system by entering

```
kill(x);
```

If you did Computer activity 54 in the same Maxima session as this activity, then Maxima will already have values stored for various terms in the sequence (x_n). This will cause an error in the next part of this activity when you try to find a closed form for x_n. To avoid this, you must first remove x from Maxima's memory using the kill command.

(ii) Solve the recurrence system by entering

```
solve_rec(x[n]=2*x[n-1]+3, x[n], x[1]=5);
```

The first argument of the solve_rec command is the recurrence relation for the sequence, entered using the syntax that you saw in Computer activity 54, with one difference, as follows.

Since you want to *solve* the recurrence system, you use an equals sign (=, which is used to specify an equation) in the recurrence relation, **not** the colon-equals sign (:=) that you used to *define* the recurrence system in Computer activity 54.

The second argument of the solve_rec command is the general term of the sequence that you want to solve for.

The third argument is the value of the initial term. If you omit this, then the closed form is given in terms of an arbitrary constant %k$_1$.

The closed form obtained should be the same as the one found in Example 12 of Unit 12.

(c) Solve the recurrence system

$$u_0 = 4, \quad u_1 = 9, \quad u_n = 5u_{n-1} - 6u_{n-2} \quad (n = 2, 3, 4, \ldots)$$

from Computer activity 54(b) by entering the following two commands.

```
kill(u);
solve_rec(u[n]=5*u[n-1]-6*u[n-2], u[n], u[0]=4, u[1]=9);
```

Since this is a second-order recurrence system, two initial terms are needed. You include both of these as the final arguments of the solve_rec command.

The closed form obtained should be the same as the one found in Example 16 of Unit 12.

The solve_rec command for solving a recurrence system is summarised below. You can practise using it in Computer activity 56.

Solving recurrence systems

Operation	Command	Example
Solve a recurrence system	`solve_rec(recurrence relation, general term, initial term(s))`	`solve_rec(a[n]=2*a[n-1], a[n],a[1]=1);`

Note: the solve_rec package needs to be loaded before this command can be used.

Computer activity 56 Solving more recurrence systems

Solve the following recurrence systems.

(a) $x_1 = 3, \quad x_n = 2x_{n-1} - 1 \quad (n = 2, 3, 4, \ldots)$

(b) $y_1 = 12, \quad y_n = -\frac{1}{3}y_{n-1} + 1 \quad (n = 2, 3, 4, \ldots)$

(c) $u_0 = 2, \quad u_1 = 7 \quad u_n = 7u_{n-1} - 12u_{n-2} \quad (n = 2, 3, 4, \ldots)$

(d) $u_0 = 2, \quad u_1 = 7 \quad u_n = 6u_{n-1} - 9u_{n-2} \quad (n = 2, 3, 4, \ldots)$

If you began studying this section from Activity 31 of Unit 12, return to the unit to continue your study.

10 Maxima accessibility guide

This section outlines issues associated with the use of Maxima if you interact with a computer using only a keyboard, use a screenreader or require different colour settings. This information should help you decide whether the wxMaxima interface is suitable for you, or whether you should consider using an alternative interface, such as the command-line interface. It also explains how to configure Maxima to make it easier for you to use and contains alternative versions of Subsection 1.2 and parts of Section 2 describing how to use the command-line interface.

Activities within this section are numbered to match corresponding activities in Sections 1 and 2. If you decide to use the command-line interface, the format of the solutions given at the end of this *Guide* may not exactly match the output you obtain.

10.1 Choosing and configuring your interface

This subsection describes how wxMaxima can be configured to help those with additional requirements. If you find these changes are not sufficient, you may wish to use the command-line interface to Maxima instead.

Using wxMaxima with a keyboard alone

The majority of the wxMaxima interface is accessible using a keyboard alone. On a Windows computer you can access the wxMaxima menus by pressing and releasing the Alt key, navigating through the menus using the keyboard arrow keys, and then pressing Enter to make your selection. Alternatively, you can press and hold down the Alt key, and then press the keyboard key that corresponds to the underlined letter in a menu entry to select it. In both cases, you can leave menu navigation without making a selection by pressing the Esc key.

On an Apple Mac you can access the wxMaxima menus by pressing and releasing the F2 key while holding down the Ctrl and Fn keys (or just Ctrl on some keyboards), navigating through the menus using the keyboard arrow keys, and then pressing Enter to make your selection. You can leave menu navigation without making a selection by pressing the Esc key.

The wxMaxima toolbar cannot be accessed using the keyboard alone, but this is not used extensively in this module and alternative typed commands for accessing the functions are given as required.

Changing wxMaxima colours and fonts

You can change the appearance of the text used by wxMaxima as follows. First, click on the following toolbar icon.

Alternatively, select `Configure` from the `Edit` menu (or, on an Apple Mac, select `Preferences` from the `wxMaxima` menu).

In the configuration window that opens, select the following ‘Style’ icon.

You can then change the colour, style, and sometimes the font and size, of individual elements of the wxMaxima interface by first choosing the appropriate element from the ‘Styles’ list, and then selecting your preferred settings.

Where a selected element has a greyed-out ‘Choose font’ button this is usually because it shares its font with other elements of the interface. For example, those elements that form part of the mathematical output of the interface share a font that you can change by first clicking on the Math font ‘Choose font’ button and then selecting your preferred font and size. Most of the remaining elements with a greyed-out font button share a font that you can change in a similar way by clicking on the Default font ‘Choose font’ button. If you choose a large font size, some horizontal scrolling may be necessary to read all the text displayed in a worksheet.

Unfortunately, the Configuration window is not keyboard accessible, and so if you only use a keyboard you may need the assistance of a non-medical helper to initially configure the system.

The Configuration window does not change the colours and fonts used in graphs produced by Maxima. Subsection 10.7 explains how to change the colours, fonts and line thickness used in graphs. You should read Subsection 10.7 in conjunction with Section 3.2, which discusses how to plot graphs.

Using a screen magnifier or screenreader

The text in wxMaxima can become pixelated when you use a screen magnifier. An alternative is to increase the size of the text font as described above and/or use the zooming facilities on the `View` menu, which you can also access using `Alt-I` (zoom in) and `Alt-O` (zoom out).

The wxMaxima interface is not accessible to a screenreader. If you use a screenreader you might wish to consider using the command-line interface to Maxima instead of the wxMaxima interface. You should ensure your screenreader is set to read all the punctuation. Details of using Maxima with the command-line interface are given below. The graphs produced by Maxima are not accessible to a screenreader. You might need assistance when using these.

10.2 Starting, testing and configuring Maxima using the command-line interface

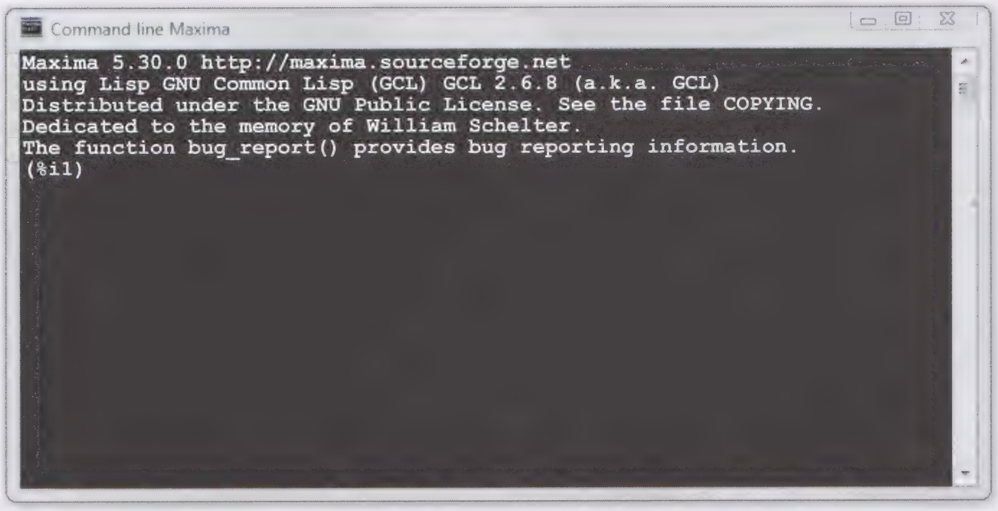
This subsection describes how to start, test and configure Maxima when using the command-line interface. If you are using the command-line interface you should read this subsection instead of Subsection 1.2, which this replaces.

Computer activity 2 Starting Maxima

Start the command-line interface to Maxima on your computer and keep it open to work with as you study this section.

If you cannot remember how to do this on your computer, see the OU Maxima website: `learn1.open.ac.uk/site/maxima`.

The command-line interface to Maxima is illustrated in Figure 12.



```
Maxima 5.30.0 http://maxima.sourceforge.net
using Lisp GNU Common Lisp (GCL) GCL 2.6.8 (a.k.a. GCL)
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
The function bug_report() provides bug reporting information.
(%i1)
```

Figure 12 The initial command-line interface window

At the top of the window is some introductory text, followed by

```
(%i1)
```

This is how Maxima labels lines. The `i` stands for 'input': this is input line number 1 and is where you can type your first command. These commands instruct Maxima to either perform calculations or change system settings.

The percentage symbol (%) is used by Maxima to indicate objects built into the system. There is more about this later.

The following activity shows you how to use Maxima to perform a simple calculation and checks that your system is working properly. Follow the steps using Maxima as you read the activity.

If you encounter unexpected problems when working through this activity, check the Frequently asked questions (FAQs) section on the OU Maxima website: `learn1.open.ac.uk/site/maxima`.

Computer activity 3 Your first Maxima calculation

Follow these steps to use Maxima to calculate $2 + 3 \times \frac{4}{5}$.

(a) Click anywhere on the Maxima window, to ensure that the text you type next is directed to the correct place.

(b) Type the following exactly as it appears here:

```
2+3*4/5;
```

including the semicolon (;) at the end.

> The text you type will appear next to the input line number (`%i1`), as indicated by a flashing horizontal line known as the **editing cursor**.

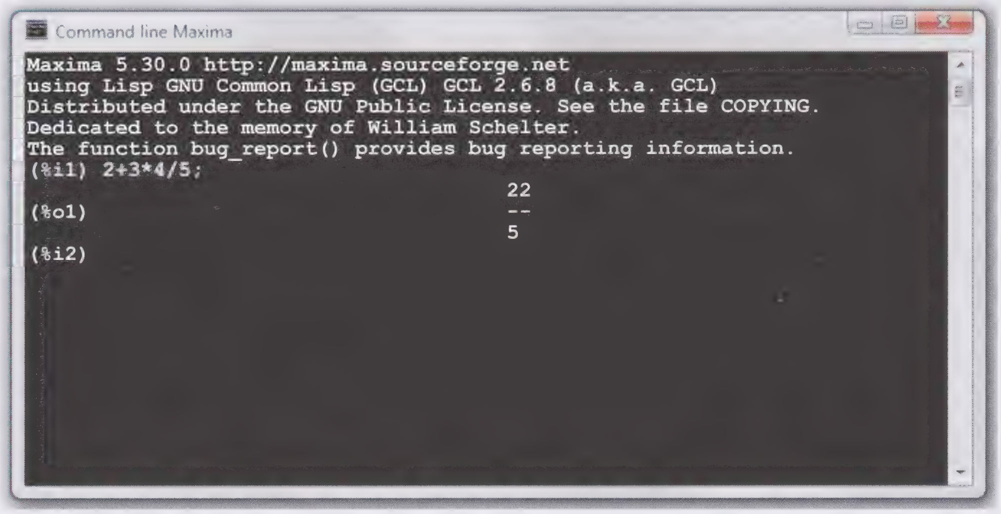
(c) Press Enter .

If you forget to type the semicolon (;) at the end of the command, the editing cursor simply moves to the next line, waiting for you to type more. Maxima will continue to do this until you indicate you have reached the end of the command by typing a semicolon. So if you forgot the semicolon, then type it now and press Enter again.

Pressing Enter instructs Maxima to calculate the expression. This is known as **evaluating** the expression.

The result is displayed under the input line, preceded by the label `(%o1)`. The `o` stands for 'output', so this label identifies the line as output line 1. It gives the output corresponding to input line 1. Throughout each Maxima session the line numbering starts at 1 and then increases by one for each new input (and output).

Maxima then displays another input line number, ready for you to enter your next command.



```
Maxima 5.30.0 http://maxima.sourceforge.net
using Lisp GNU Common Lisp (GCL) GCL 2.6.8 (a.k.a. GCL)
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
The function bug_report() provides bug reporting information.
(%i1) 2+3*4/5;
                                       22
(%o1)                                  --
                                       5
(%i2)
```

If, in Computer activity 3, Maxima behaved as described, then you have a working system to use throughout MST125.

The result of the calculation in Computer activity 3 was displayed as formatted mathematics over several lines, replicating how mathematics is usually written. This can, however, cause problems if you are using a screenreader. The display behaviour can be changed for every Maxima session as described in the following activity.

Computer activity 4 Configuring the Maxima display of mathematics

You should only do this activity if you wish to change how Maxima displays mathematics.

(a) First exit Maxima by entering the command

```
quit();
```

(on some installations you may also have to close the window).

(b) Turn off the two-dimensional display of mathematics by adding the Maxima command `display2d:false;` to your `maxima-init.mac` configuration file, as follows.

(i) Follow the instructions in the Accessibility section of the OU Maxima website `learn1.open.ac.uk/site/maxima` to locate your `maxima-init.mac` configuration file, or create one in the correct location as appropriate.

(ii) Open your `maxima-init.mac` file in a plain text editor (such as Notepad on a Windows computer, or TextEdit on an Apple Mac), and add the text

```
display2d:false;
```

on a new line. There are further details on how to do this in the Accessibility section of the OU Maxima website.

(iii) Save your modified `maxima-init.mac`, and close your text editor.

(c) Restart Maxima and enter the calculation from Computer activity 3 again, that is, enter

```
2+3*4/5;
```

Notice that Maxima now displays the result on a single line:

```
22/5
```

Another useful configuration change is to make Maxima display the result of a command left-justified against the output line number, instead of the default behaviour of displaying it centred on the line. This can be achieved by entering the command

```
leftjust:true;
```

either in the Maxima window, or, if you wish to make the change permanent, in your `maxima-init.mac` file as described above.

Closing wxMaxima

When you have finished using Maxima, close it by using one of the following methods.

- Enter `quit();`
- On a Windows computer, click the usual small cross button at the top right-hand corner of the Maxima window.

10.3 Troubleshooting Maxima using the command-line interface

If you are using the command-line interface to Maxima and it does not seem to be responding, try the following suggestions.

Troubleshooting Maxima

- Check you have ended your previous command with either a semicolon (`;`) or a dollar (`$`). Maxima will not do anything without them!
- Check that Maxima is not waiting for you to type something. Is a question being displayed that you have not yet answered?
- If Maxima is taking too long to evaluate a command, you can interrupt it by pressing `Ctrl-C`.

  If you do this, you may see the following output, or something similar, which you can safely ignore.

```
Maxima encountered a Lisp error:

  EXT:GC: User break

Automatically continuing.

To enable the Lisp debugger set *debugger-hook* to nil.
```

Note that pressing `Ctrl-C` at other times may close your Maxima session.

If you encounter other problems when using Maxima, check the Frequently asked questions (FAQs) section of the OU Maxima website: `learn1.open.ac.uk/site/maxima`.

10.4 Reusing and editing commands in the command-line interface

When using the command-line interface, you can edit and re-evaluate previously entered commands or calculations, as shown in the following activity.

Computer activity 7 Editing a command

(a) Calculate $2\sqrt{3}+5\sqrt{27}$ by entering the following line into Maxima.

```
2*sqrt(3)+5*sqrt(27);
```

(b) Edit your entry in part (a) to calculate $2\sqrt{3}-5\sqrt{27}$ as follows.

(i) Press the keyboard up arrow key once, so that the previously entered command appears against the current input line number (on an Apple Mac you may need to hold down the Alt key while pressing the keyboard up arrow key once).

You can access all the commands you have previously entered in your Maxima session using the up and down keyboard arrow keys.

(ii) Edit the expression by moving the editing cursor using the left and right keyboard arrow keys, typing new text and using the ⟵ (backspace) key to make deletions.

(iii) Press Enter .

Maxima also permits the result of one calculation to be used in another calculation. The symbol `%` represents the result of the last evaluated command. So, for example, if you enter `sqrt(%);` Maxima calculates the square root of the last evaluated output. You can also use the output (or input) of other lines, by typing the line label in a calculation. For example, if you enter `%o5^3;` Maxima calculates the cube of the expression in output line 5.

Computer activity 8 Using the results of previous calculations

(a) Enter `sin(5*%pi/6);` into Maxima and evaluate it.

(b) On the following line type `2*%;` and press Enter .

Maxima returns `1`, which is twice the previous answer.

(c) Enter `%o▨^2;` with ▨ replaced by the output line number corresponding to the result of `sin(5*%pi/6);` entered earlier.

This command finds the square of the result of the line referenced.

10.5 Saving and printing your work using the command-line interface

After working with Maxima you will probably want to save your calculations so that you can reuse or read them at a later date.

When you are using the command-line interface you cannot save the session directly, as you do when saving other files. Instead there are two stages in saving your work:

- If you want to keep a record of the contents of the command-line window, then you need to save a *transcript* of the session. You can then print this transcript if needed.
- If you want to save the values of any variables and functions you have defined, so that you can continue working with them in a new Maxima session, then you will need to save the current state of the system. Note that this does not give you access to the commands used to define the variables or functions, so you cannot edit them in the new session.

These two processes are illustrated in Computer activity 9.

When saving your work, you need to tell Maxima exactly where to save the file containing the session state or your transcript by giving the complete *pathname* of the file, which includes not only the name of the file but also the hierarchy of folders in which the file is located. Unless you do this, Maxima will try to save the file in the location in which the system is installed on your computer. Not only is this not recommended, but as a computer user, you may not have sufficient permission to do this.

The pathname of a file depends on its location, the type of computer you are using, and your computer username. In the following activity, we assume that a user with the computer username ‘bill’ wishes to save a transcript of his Maxima session in a file called `transcript.txt` within a folder called `maxima` which he created in Computer activity 1. The `maxima` folder is within Bill’s ‘Documents’ area on a recent Windows computer (that is, Windows Vista, 7, 8 or 10). In this case, the full pathname of his file is

```
C:/Users/bill/Documents/maxima/transcript.txt
```

(Notice that Maxima separates parts of the pathname with a forward slash (`/`) rather than the usual Windows backward slash (\).)

If Bill were using an older Windows computer (such as Windows XP), the equivalent pathname would be

```
C:/Documents and Settings/bill/Documents/maxima/transcript.txt
```

Using an Apple Mac the equivalent is

```
/Users/bill/Documents/maxima/transcript.txt
```

Computer activity 9 Saving your work

(a) Create a transcript of your current session by following these steps.

(i) Enter `writefile("C:/Users/bill/Documents/maxima/transcript.txt");` replacing the pathname with one appropriate to you and your system.

The path of the file is given in double quote marks.

The command tells Maxima to save a copy of all the subsequent input and outline lines to the file named `transcript.txt` in your chosen folder.

The output

```
#<OUTPUT BUFFERED FILE-STREAM CHARACTER
C:\Users\bill\Documents\maxima\transcript.txt>
done
```

is given, where the pathname is the one you gave in the command.

(ii) Enter

```
playback();
```

This command redisplays each of the input and output lines of your session, and when used in conjunction with the `writefile` command results in a copy of these lines being stored in your chosen file.

To playback only a selection of the lines from your session, you can use a command of the form

```
playback([first line number, last line number]);
```

which plays back only those lines within the given range.

(iii) Enter

```
closefile();
```

This stops the recording of the session into the file.

The output

```
#<CLOSED OUTPUT BUFFERED FILE-STREAM CHARACTER
C:\Users\bill\Documents\maxima\transcript.txt>
done
```

is given.

(iv) Find the file `transcript.txt` on your computer and open it using a text editor, for example, Microsoft Word.

The file contains a copy of the input and output lines from your session. (When opened in some text editors the file may appear all on one long line.)

(b) Save your current session then reload it, by following these steps.

(i) Enter

```
save("C:/Users/bill/Documents/maxima/mysession.lisp", all);
```

replacing the pathname with one appropriate to you and your system.

> This tells Maxima to save your current session to the file `mysession.lisp` in your chosen folder. (Lisp is the computer programming language used to create Maxima, and the `save` command saves your work using commands from this language.)
>
> The path of the file is output.

(ii) Close Maxima by entering

```
quit();
```

(iii) Restart Maxima in the usual way.

(iv) Load your previously saved session by entering

```
load("C:/Users/bill/Documents/maxima/mysession.lisp");
```

using the same pathname you used in part (b)(i).

> This tells Maxima to restore the state of the system to that saved in the specified file.

(v) Enter

```
values;
```

to display the names of all known variables.

> You should see that all the variables from your previous session, if any, are known. You might like to check their values are as expected too.
>
> There is more about variables and the `values` command in Subsection 2.6.

An alternative way of using the `writefile` and `closefile` commands is to enter the first at the very start of your session and the second at the end, so that everything you do is recorded.

Saving your work

Operation	Command
Start writing a transcript	`writefile("file path");`
Stop writing a transcript	`closefile();`
Playback all the input and output in a session	`playback();`
Playback a range of input and output	`playback([start line number, end line number]);`
Save the state of a session	`save("file path", all);`
Reload a session	`load("file path");`

You can also obtain an image of the Maxima window by taking a 'screenshot'. To do this on a Microsoft Windows computer, first click on the Maxima window, then press `Alt-PrintScreen` (that is, hold down the Alt keyboard key while pressing the `PrintScreen` key, which may be labelled PrtScr, Prt Scrn or something similar). This stores the image of the Maxima window, which you can then paste into a suitable application. If you are using Windows Vista, Windows 7, 8 or 10, you might like to use the 'Snipping Tool', available by typing 'snip' into the Start menu.

To take a screenshot on an Apple Mac computer, press `Cmd-Ctrl-Shift-4` (that is, hold down the Command, Ctrl and ⇧ keyboard keys while pressing the 4 key). Next press the space bar, move the camera pointer over the Maxima window and click. This stores the image of the Maxima window, which you can then paste into a suitable application. You may also like to use the Grab application from the Utilities folder.

10.6 Plotting graphs without wxMaxima

When using the command-line interface, you should use the `plot2d` command rather than the `wxplot2d` command described in Section 3.2. These commands work similarly, and have the same arguments. The main difference is that `plot2d` opens a new computer window containing the graph.

The graph window can be closed by pressing the usual cross at the top right-hand corner, or by pressing `Alt-F4`. You will not be able to enter any commands into the Maxima command-line interface until the graphing window is closed.

The graph can be saved as a `.png`, `.pdf` or `.svg` file using the save button. These files can be opened with many image viewing programs and can be inserted into word-processed documents.

10.7 Changing properties of graphs

You can change the thickness of curves plotted in graphs by including an additional argument in the `wxplot2d` command, of the form

```
[style, [lines, 3]]
```

where the number indicates how thick the line should be. By default, lines have thickness 1.

So, to plot a graph of the equation $y = 2x$ with a thicker line, enter, for example,

```
wxplot2d(2*x, [x, 0, 1], [style, [lines, 3]]);
```

Typing this additional argument each time you want to plot a graph can be cumbersome, so you might like to assign this argument to a variable that you can reuse. For example, to assign your preferred style to the variable `thick` enter

```
thick:  [style, [lines, 3]];
```

Then you can plot graphs using commands such as

```
wxplot2d(2*x, [x, 0, 1], thick);
```

To change the colour of a curve, you can include an argument of the form `[color, red]` as described in Subsection 3.2. Alternatively, you can add a further element to the styles argument described above. For example, including

```
[style, [lines, 3, 5]]
```

produces curves with thickness 3 and colour number 5, which is black. The colour number codes used by the `style` argument are given in the table below.

`style` argument colour numbers

Number	Colour
1	blue
2	red
3	green
4	magenta
5	black
6	cyan

Note that the value of any variable you define to help set curve properties (such as `thick` above) will be lost when you close Maxima at the end of a session unless you save your session. The `style` argument also only changes properties of the curve plotted, not other aspects of the graph such as the axes and labels.

Changing the axes and labels, and ensuring such changes are automatically used each time Maxima is started, is more complicated. A computer file is provided in the Accessibility section of the OU Maxima website `learn1.open.ac.uk/site/maxima` which you can download and configure to set the graph properties to your needs. These settings will then be automatically used for each graph in every Maxima session. Details of how to download and use this are given in the Accessibility section of the OU Maxima website.

Computer methods for CAS Activities in Books A–D

Computer solution to Unit 4, Activity 22

First, load the `implicit_plot` package to enable the use of the `wximplicit_plot` command for plotting curves given by equations in implicit form.

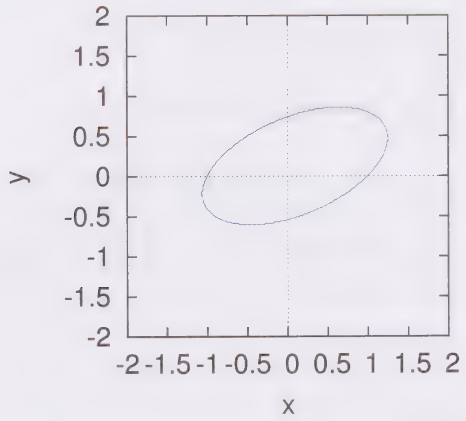
```
(%i1) load(implicit_plot);
(%o1) C:\maxima-5.38.1\share\maxima\5.38.1_5\share\contrib\
       implicit_plot.lisp
```

(The output from this command is the location of the package on your computer system, and may differ from that shown here.)

(a) The curve represented by $2x^2 - 3xy + 5y^2 - y - 2 = 0$ can be plotted as follows.

```
(%i2) wximplicit_plot(2*x^2-3*x*y+5*y^2-y-2=0,
      [x,-2,2], [y,-2,2], same_xy);
```

(%t2)



(%o2)

Here, axis ranges have been chosen to ensure the whole curve is visible and the axes have been set to have equal scales.

If the curve in your plot is not fully visible, or is too small, adjust the ranges of your axes until a reasonable figure is obtained.

The curve appears to be an ellipse.

The equation

$$2x^2 - 3xy + 5y^2 - y - 2 = 0$$

is of the form $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$ with $A = 2$, $B = -3$ and $C = 5$. This gives

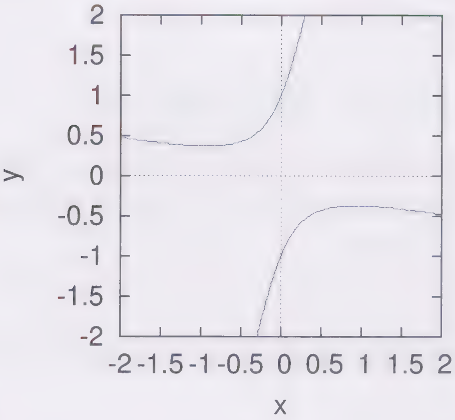
$$B^2 - 4AC = (-3)^2 - 4 \times 2 \times 5 = 9 - 40 = -31.$$

Since $B^2 - 4AC < 0$, and $A \neq C$, the conic is indeed an ellipse.

(b) The curve represented by $x^2 + 5xy - y^2 + 1 = 0$ can be plotted as follows.

```
(%i3) wximplicit_plot(x^2+5*x*y-y^2+1=0,
      [x,-2,2], [y,-2,2], same_xy);
```

(%t3)



(%o3)

The curve appears to be a hyperbola.

The equation

$$x^2 + 5xy - y^2 + 1 = 0$$

is of the form $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$ with $A = 1$, $B = 5$ and $C = -1$. This gives

$$B^2 - 4AC = 5^2 - 4 \times 1 \times (-1) = 25 + 4 = 29.$$

Since $B^2 - 4AC > 0$, the conic is indeed a hyperbola.

Computer solution to Unit 4, Activity 35

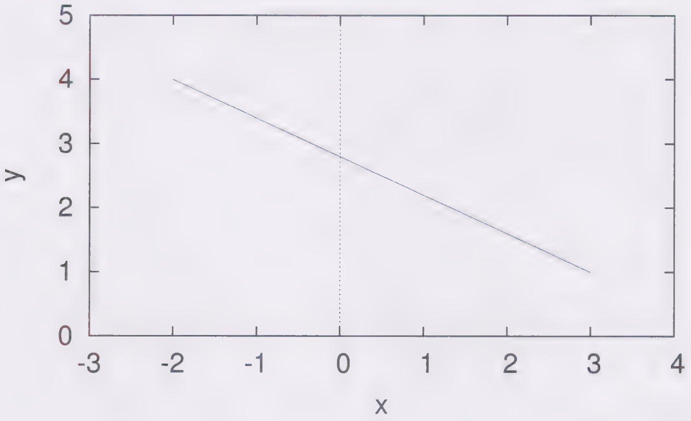
(a) The parametrisation of the line joining $(-2, 4)$ and $(3, 1)$ found in the solution to Activity 33(b) of Unit 4 is

$$x = -2 + 5t, \quad y = 4 - 3t \quad (0 \leq t \leq 1).$$

This can be plotted as follows.

```
(%i1) wxplot2d([parametric, -2+5*t, 4-3*t, [t,0,1]],
       [x,-3,4], [y,0,5], [xlabel,"x"], [ylabel,"y"]);
```

(%t3)



(%o3)

Here, axis ranges have been chosen to clearly show the ends of the line segment, and suitable axis labels added.

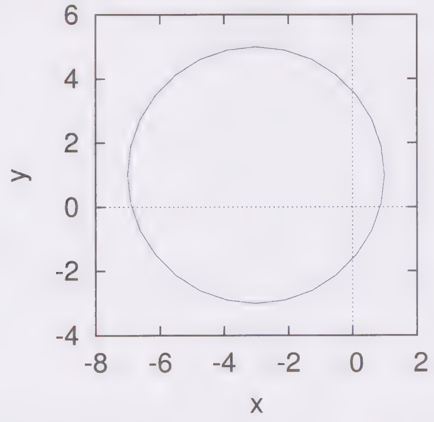
(b) The parametrisation of the circle with centre $(-3, 1)$ and radius 4 found in the solution to Activity 34(a) of Unit 4 is

$$x = -3 + 4\cos t, \quad y = 1 + 4\sin t \quad (0 \leq t \leq 2\pi).$$

This can be plotted as follows.

```
(%i2) wxplot2d([parametric, -3+4*cos(t), 1+4*sin(t), [t,0,2*%pi]],
       [x,-8,2], [y,-4,6], [xlabel,"x"], [ylabel,"y"], same_xy);
```

(%t3)



(%o3)

Here, axis ranges have been chosen to clearly show the entire curve, and suitable axis labels added. Since a circle is being plotted, equally scaled axes have been used.

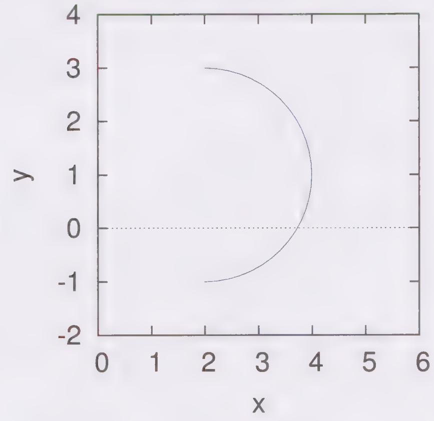
(c) The parametrisation of the semicircle in Activity 34(b) of Unit 4 found in the solution to this activity is

$$x = 2 + 2\cos t, \quad y = 1 + 2\sin t \quad \left(-\frac{\pi}{2} \leq t \leq \frac{\pi}{2}\right).$$

This can be plotted as follows.

```
(%i3) wxplot2d([parametric, 2+2*cos(t), 1+2*sin(t),
       [t,-%pi/2,%pi/2]],[x,0,6], [y,-2,4],
       [xlabel,"x"], [ylabel,"y"], same_xy);
```

(%t3)



(%o3)

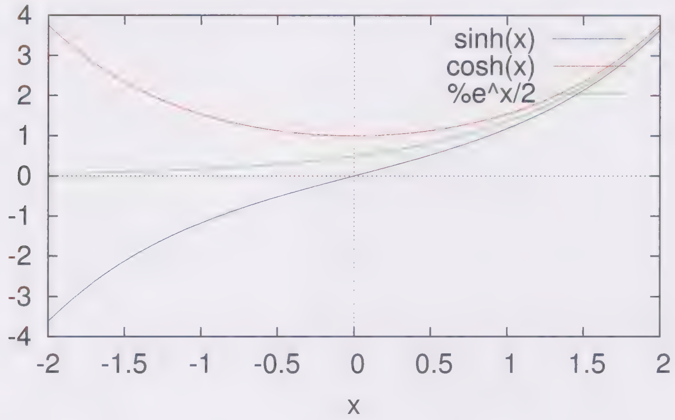
Here, axis ranges have been chosen to clearly show the entire curve, and suitable axis labels added. Equally scaled axes have been used.

Computer solution to Unit 7, Activity 37

(a) From the 'Maxima reference guide', the commands for $\sinh x$ and $\cosh x$ are `sinh(x)` and `cosh(x)`, respectively. So the graphs of $\sinh x$, $\cosh x$ and $\frac{1}{2}e^x$ can be plotted as follows.

```
(%i1) wxplot2d([sinh(x), cosh(x), exp(x)/2], [x,-2,2]);
```

(%t3)



(%o3)

Alternatively, since $\sinh x = \frac{1}{2}(e^x - e^{-x})$ and $\cosh x = \frac{1}{2}(e^x + e^{-x})$, the graphs of $\sinh x$, $\cosh x$ and $\frac{1}{2}e^x$ can be plotted, with an appropriate legend, using

```
wxplot2d([(exp(x)-exp(-x))/2, (exp(x)+exp(-x))/2, exp(x)/2],
[x,-2,2], [legend,"sinh x","cosh x","exp(x)/2"]);
```

(b) When x is positive and of large magnitude, the expression e^{-x} is close to 0, and e^x is positive with large magnitude. So $\sinh x$ and $\cosh x$ are both positive with large magnitude, and both close in value to $\frac{1}{2}e^x$.

(c) When x is negative and of large magnitude, the expression e^x is close to 0, and e^{-x} is positive with large magnitude. So $\sinh x$ is negative with large magnitude, and is close in value to $-\frac{1}{2}e^{-x}$, whereas $\cosh x$ is positive with large magnitude, and is close in value to $\frac{1}{2}e^{-x}$.

Solutions to Computer activities

Solution to Computer activity 3

```
(%i1) 2+3*4/5;
```

(%o1) $\frac{22}{5}$

So $2 + 3 \times \frac{4}{5} = \frac{22}{5}$.

Solution to Computer activity 5

(a)

```
(%i1) 32^(1/5);
(%o1) 2
```

So $\sqrt[5]{32} = 2$. Note that Maxima returns the real fifth root of 32.

(b)

```
(%i2) sin(%pi/3);
```

(%o2) $\frac{\sqrt{3}}{2}$

So $\sin\left(\frac{\pi}{3}\right) = \frac{\sqrt{3}}{2}$.

(c)

```
(%i3) tan(30*%pi/180);
```

(%o3) $\frac{1}{\sqrt{3}}$

So $\tan 30^\circ = \frac{1}{\sqrt{3}}$.

(d)

```
(%i4) %e^4;
```

(%o4) $\%e^4$

(e)

```
(%i5) log(3.4);
(%o5) 1.223775431622116
```

So $\ln 3.4 \approx 1.223\,775\,431\,622\,116$.

Do not be surprised if your line numbering differs from ours. Each of our solutions is prepared as a separate Maxima session with the line numbering starting at 1 (unless it is a continuation of a previous activity); your numbering will continue to increase for the duration of your session.

Solution to Computer activity 6

(a)

```
(%i1) sqrt(325);
```

(%o1) $5\sqrt{13}$

```
(%i2) float(sqrt(325));
(%o2) 18.02775637731995
```

So $\sqrt{325} = 5\sqrt{13} \approx 18.027\,756\,377\,319\,95$.

(b) `(%i3) 5^200;`

`(%o3) 622301527786114170714406405378[80digits]`
`551224821247160434722900390625`

`(%i4) float(5^200);`

`(%o4)` $6.2230152778611415\ 10^{139}$

So $5^{200} \approx 6.223\,015\,277\,861\,1415 \times 10^{139}$.

Your answers may differ in the last decimal place or so to those given here. This is due to small differences in the way floating-point numbers are computed by different systems.

Solution to Computer activity 7

(a) `(%i1) 2*sqrt(3)+5*sqrt(27);`

`(%o1)` $17\sqrt{3}$

So $2\sqrt{3} + 5\sqrt{27} = 17\sqrt{3}$.

(b) `(%i2) 2*sqrt(3)-5*sqrt(27);`

`(%o2)` $-13\sqrt{3}$

So $2\sqrt{3} - 5\sqrt{27} = -13\sqrt{3}$.

Solution to Computer activity 8

At the end of this activity, your worksheet should look like the following, although line numbers may differ.

`(%i4) sin(5*%pi/6);`

`(%o4)` $\frac{1}{2}$

`(%i5) 2*%;`

`(%o5) 1`

`(%i6) %o4^2;`

`(%o6)` $\frac{1}{4}$

Solution to Computer activity 9

(a) `(%i1) a:sqrt(8);`

`(%o1)` $2^{3/2}$

(b) `(%i2) a;`

`(%o2)` $2^{3/2}$

(c) `(%i3) b:a*sqrt(2);`

`(%o3) 4`

(d)
```
(%i4) a:sqrt(7);
(%o4) √7
```

(e)
```
(%i5) b;
(%o5) 4
```

(f)
```
(%i6) values;
(%o6) [a, b]
```

(g)
```
(%i7) kill(a);
(%o7) done
```

(h)
```
(%i8) a;
(%o8) a
```

(i)
```
(%i9) b;
(%o9) 4
```

Solution to Computer activity 10

(a)
```
(%i1) a:42;
(%o1) 42
```

(b)
```
(%i2) b:sin(a)^2+cos(a)^2;
(%o2) sin(42)^2 + cos(42)^2
```

(c)
```
(%i3) float(b);
(%o3) 1.0
```

Solution to Computer activity 11

(a)
```
(%i1) fpprintprec:8;
(%o1) 8
```

(This sets the number of significant figures to be shown in the display of decimal numbers.)

(b)
```
(%i2) float(%pi);
(%o2) 3.1415927
```

So the value of π is 3.141 5927 (to 8 s.f.).

Solution to Computer activity 12

The circumference of a circle of radius 5 is 31.42 (to 2 d.p.) and the area is 78.54 (to 2 d.p.).

Solution to Computer activity 14

(a)

```
(%i1) ? float;

-- Function: float (<expr>)
     Converts integers, rational numbers and bigfloats in
     <expr> to floating point numbers.  It is also an
     ‘evflag’, ‘float’ causes non-integral rational numbers
     and bigfloat numbers to be converted to floating point.
  There are also some inexact matches for ‘float’.
  Try ‘?? float’ to see them.
(%o1) true
```

(b)

```
(%i2) ?? sqr;

 0: isqrt  (Functions and Variables for Number Theory)
 1: sqrt  (Root Exponential and Logarithmic Functions)
 2: sqrtdenest  (Package sqdnst)
 3: sqrtdispflag  (Functions and Variables for Display)
--> 1;

-- Function: sqrt (<x>)
     The square root of <x>.  It is represented internally by
     ‘<x>^(1/2)’.  See also ‘rootscontract’ and ’radexpand’.
(%o2) true
```

Solution to Computer activity 15

(a)

```
(%i1) p:2*x+sqrt(1+3*x^6);
```

(%o1) $\sqrt{3x^6+1}+2x$

(b)

```
(%i2) subst(3,x,p);
```

(%o2) $2\sqrt{547}+6$

So, when $x = 3$, $2x + \sqrt{1+3x^6} = 2\sqrt{547}+6$.

(c)

```
(%i3) f(x):=(x^4+x+1)/(x^3-13*x+12);
```

(%o3) $f(x) := \dfrac{x^4+x+1}{x^3-13x+12}$

(d)

```
(%i4) f(2);
```

(%o4) $-\dfrac{19}{6}$

So $f(2) = -\dfrac{19}{6}$.

(e)

```
(%i5) f(3);

expt: undefined: 0 to a negative exponent.
#0: f(x=3)
-- an error. To debug this try: debugmode(true);
```

(f)
```
(%i4) g(1);
(%o4) g(1)
```

(g)
```
(%i5) functions;
(%o5) [f(x)]
```

Solution to Computer activity 16

(a)
```
(%i1) expand( (1+sin(x))*(x+4)^2 );
```
(%o1) $\mathtt{x}^2\mathtt{sin}(\mathtt{x}) + 8\,\mathtt{x}\,\mathtt{sin}(\mathtt{x}) + 16\,\mathtt{sin}(\mathtt{x}) + \mathtt{x}^2 + 8\,\mathtt{x} + 16$

So $(1+\sin x)(x+4)^2 = x^2\sin(x) + 8x\sin(x) + 16\sin(x) + x^2 + 8x + 16$.

(b)
```
(%i2) factor( 3*x^3+11*x^2+8*x-4 );
```
(%o2) $(\mathtt{x}+2)^2(3\mathtt{x}-1)$

So $3x^3 + 11x^2 + 8x - 4 = (x+2)^2(3x-1)$.

(c)
```
(%i3) fullratsimp( (2*x^3-3*x^2+1)/(x-1) );
```
(%o3) $2\mathtt{x}^2 - \mathtt{x} - 1$

So $\dfrac{2x^3 - 3x^2 + 1}{x-1} = 2x^2 - x - 1$.

(d)
```
(%i4) radcan( log(x^3-3*x-2)-2*log(x+1) );
```
(%o4) $\mathtt{log}(\mathtt{x}-2)$

So $\ln(x^3 - 3x - 2) - 2\ln(x+1) = \ln(x-2)$.

You might like to think about how you could obtain the same answer by hand.

(e)
```
(%i5) logcontract( log(2*x)-2*log(x)+log(y) );
```
(%o5) $\mathtt{log}\left(\dfrac{2\mathtt{y}}{\mathtt{x}}\right)$

So $\ln(2x) - 2\ln x + \ln y = \ln\left(\dfrac{2y}{x}\right)$.

(f)
```
(%i6) trigexpand( tan(3*x) );
```
(%o6) $\dfrac{3\,\mathtt{tan}(\mathtt{x}) - \mathtt{tan}(\mathtt{x})^3}{1 - 3\,\mathtt{tan}(\mathtt{x})^2}$

So $\tan(3x) = \dfrac{3\tan x - \tan^3 x}{1 - 3\tan^2 x}$.

(g)
```
(%i7) trigreduce( cos(x)^4 );
```
(%o7) $\dfrac{\mathtt{cos}(4\mathtt{x}) + 4\,\mathtt{cos}(2\mathtt{x}) + 3}{8}$

So $\cos^4 x = \dfrac{\cos(4x) + 4\cos(2x) + 3}{8}$.

(h)
```
(%i8) trigsimp( cos(x)^2*(1+tan(x)^2) );
(%o8) 1
```

So $\cos^2 x\left(1+\tan^2(x)\right)=1$.

(i)
```
(%i9) trigrat( tan(x)/sin(2*x) );
```

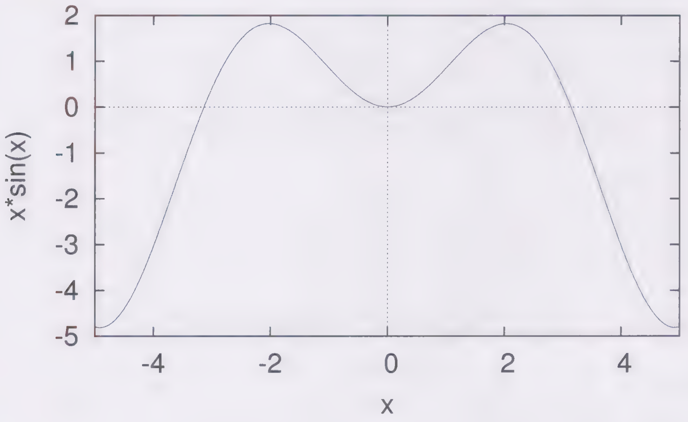
(%o9) $\dfrac{1}{\cos(2x)+1}$

So $\dfrac{\tan x}{\sin(2x)}=\dfrac{1}{\cos(2x)+1}$.

Solution to Computer activity 17

(a)
```
(%i1) wxplot2d(x*sin(x), [x,-5,5]);
```

(%t1)



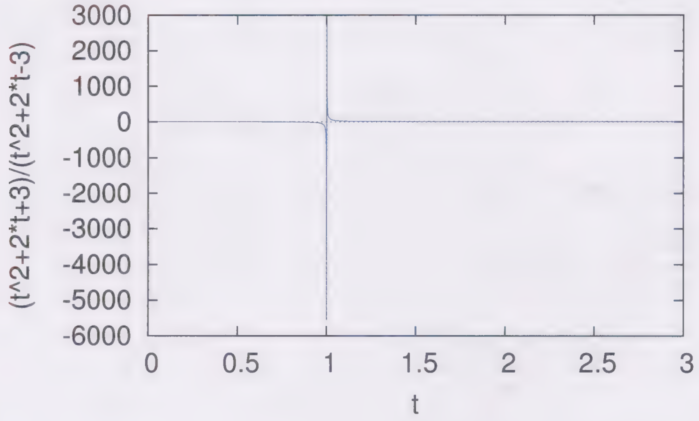
```
(%o1)
```

(b)
```
(%i2) f(t):=(t^2+2*t+3)/(t^2+2*t-3);
```

(%o2) $\mathrm{f}(t) := \dfrac{t^2+2t+3}{t^2+2t-3}$

```
(%i3) wxplot2d(f(t), [t,0,3]);
```

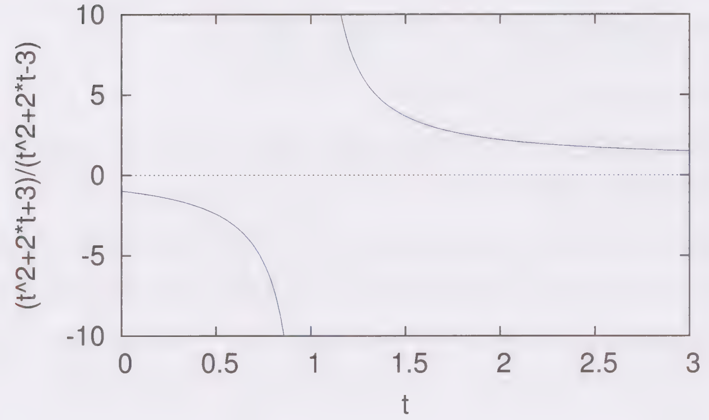
(%t3)



```
(%o3)
```

(c)
```
(%i4) wxplot2d(f(t), [t,0,3], [y,-10,10]);
plot2d:  some values were clipped.
```

(%t4)



(%o4)

Solution to Computer activity 18

(a)

```
(%i1) p(x):=x*cos(x);
```

(%o1) $p(x) := x\cos(x)$

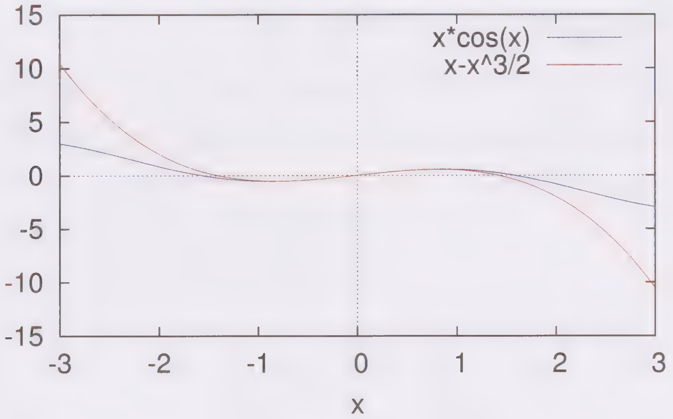
```
(%i2) q(x):=x-x^3/2;
```

(%o2) $q(x) := x - \dfrac{x^3}{2}$

(b)

```
(%i3) wxplot2d([p(x),q(x)],[x,-3,3]);
```

(%t3)

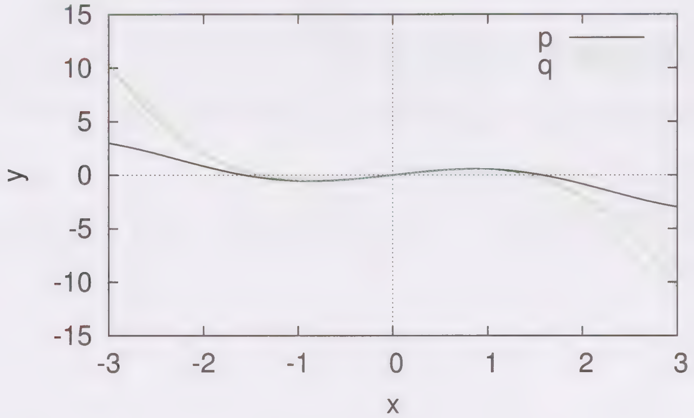


(%o3)

Solution to Computer activity 19

```
(%i4) wxplot2d([p(x),q(x)], [x,-3,3], [color, black, green],
       [legend, "p", "q"], [ylabel, "y"]);
```

(%t4)



(%o4)

Solution to Computer activity 20

(a)

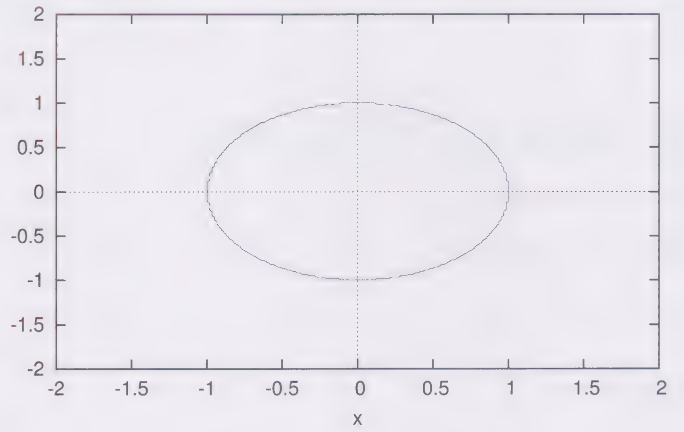
```
(%i1) load(implicit_plot);
(%o1) C:\maxima−5.38.1\share\maxima\5.38.1_5\share\contrib\
      implicit_plot.lisp
```

(The output from this command is the location of the package on your computer system, and may differ from that shown here.)

(b)

```
(%i2) wximplicit_plot(x^2+y^2=1, [x,-2,2], [y,-2,2]);
```

(%t2)



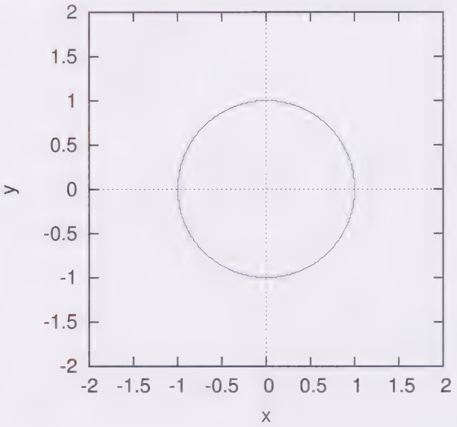
(%o2)

Solution to Computer activity 21

(It is assumed the `implicit_plot` package has already been loaded.)

```
(%i1) wximplicit_plot(x^2+y^2=1, [x,-2,2], [y,-2,2], same_xy);
```

(%t1)



(%o1)

Solution to Computer activity 22

(It is assumed the `implicit_plot` package has already been loaded.)

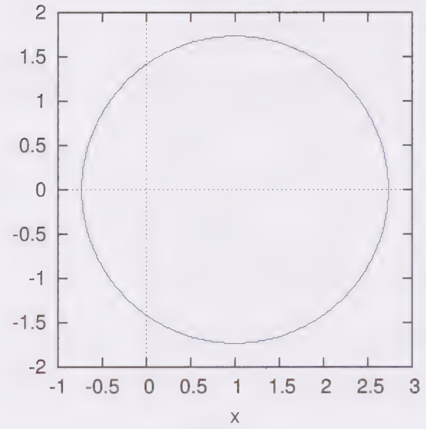
(a)

```
(%i1) set_plot_option(same_xy)$
```

(b)

```
(%i2) wximplicit_plot((x-1)^2+y^2=3, [x,-1,3], [y,-2,2]);
```

(%t2)



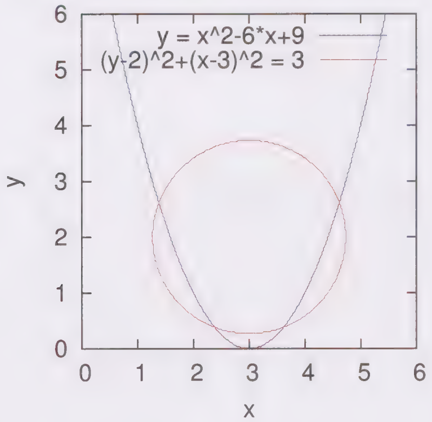
(%o2)

Solution to Computer activity 23

(It is assumed the `implicit_plot` package has already been loaded, and the plot settings changed to use equal scales for the x- and y-axes.)

```
(%i1) wximplicit_plot([y=x^2-6*x+9, (x-3)^2+(y-2)^2=3],
       [x,0,6], [y,0,6]);
```

(%t1)



(%o1)

Solution to Computer activity 24

(a)

```
(%i1) solve( x^3-4*x^2+2*x+4=0 );
```

(%o1) $[x = 1 - \sqrt{3}, x = \sqrt{3} + 1, x = 2]$

So the three solutions are $x = 1 - \sqrt{3}$, $x = 1 + \sqrt{3}$ and $x = 2$.

(b)

```
(%i2) eqn:y=log(u*x+v);
```

(%o2) $y = \log(u\,x + v)$

(c)

```
(%i3) solve(eqn, x);
```

(%o3) $[x = \frac{\%e^{y} - v}{u}]$

So the rearranged equation is $x = \dfrac{e^y - v}{u}$.

(d) `(%i4) solve(tan(3*x-1)=sqrt(3));`

```
solve:  using arc-trig functions to get a solution.
Some solutions will be lost.
```

(%o4) $[x = \frac{\pi + 3}{9}]$

So one solution of the equation is $x = \frac{\pi + 3}{9}$.

(e) `(%i5) solve(x^2-4*x+20=0);`

(%o5) $[x = 2 - 4\%i, x = 4\%i + 2]$

The two complex solutions are $x = 2 - 4i$ and $x = 2 + 4i$.

(f) `(%i6) solve(x^9+2*x-1=0);`

(%o6) $[0 = x^9 + 2x - 1]$

Maxima was unable to solve this equation.

Solution to Computer activity 25

(a) `(%i1) solns:solve(x^2-x-1=0);`

(%o1) $[x = -\frac{\sqrt{5} - 1}{2}, x = \frac{\sqrt{5} + 1}{2}]$

So the solutions are $x = \frac{1 - \sqrt{5}}{2}$ and $x = \frac{1 + \sqrt{5}}{2}$.

(b) `(%i2) solns[1];`

(%o2) $x = -\frac{\sqrt{5} - 1}{2}$

(c) `(%i3) rhs(solns[1]);`

(%o3) $-\frac{\sqrt{5} - 1}{2}$

(d) `(%i4) rhs(solns[2]);`

(%o4) $\frac{\sqrt{5} + 1}{2}$

Solution to Computer activity 26

(a) (i) `(%i1) eq1:3*x+2*y=1;`

(%o1) $2y + 3x = 1$

`(%i2) eq2:5*x+3*y=3;`

(%o2) $3y + 5x = 3$

(ii) `(%i3) solve( [eq1,eq2], [x,y] );`

(%o3) $[[x = 3, y = -4]]$

The solution of the equations is $x = 3$, $y = -4$.

(b)

```
(%i4) solve([y=x^2-6*x+9,(x-3)^2+(y-2)^2=3],[x,y]);
```

(%o4) $$[[x=-\frac{\sqrt{5}-5}{2},y=\frac{\sqrt{5}+3}{2}],[x=\frac{\sqrt{5}+5}{2},y=-\frac{\sqrt{5}-3}{2}],$$
$$[x=-\frac{\sqrt{5}-7}{2},y=-\frac{\sqrt{5}-3}{2}],[x=\frac{\sqrt{5}+7}{2},y=\frac{\sqrt{5}+3}{2}]]$$

So the solutions (the points of intersection) are
$x=\frac{5-\sqrt{5}}{2}, y=\frac{3+\sqrt{5}}{2}$; $x=\frac{5+\sqrt{5}}{2}, y=\frac{3-\sqrt{5}}{2}$;
$x=\frac{7-\sqrt{5}}{2}, y=\frac{3-\sqrt{5}}{2}$ and $x=\frac{7+\sqrt{5}}{2}, y=\frac{3+\sqrt{5}}{2}$.

Solution to Computer activity 27

(a) (i)

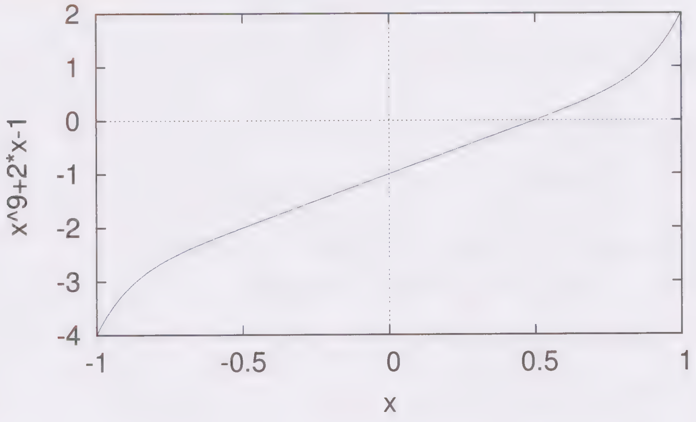
```
(%i1) f(x):=x^9+2*x-1;
```

(%o1) $f(x) := x^9 + 2x - 1$

(ii)

```
(%i2) wxplot2d(f(x), [x,-1,1]);
```

(%t2)



(%o2)

Since f(0)<0 and f(1)>0, there must be a solution in the interval $(0,1)$.

(iii)

```
(%i3) find_root(f(x), x, 0, 1);
(%o3) 0.499040180356976
```

So a solution of $x^9+2x-1=0$ is 0.499 (to 3 s.f.).

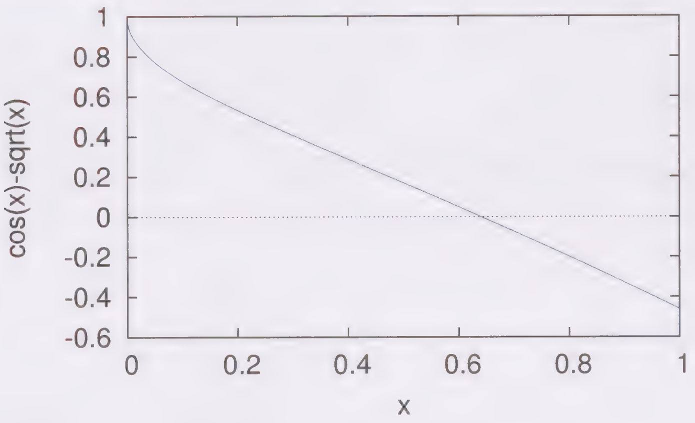
(b)

```
(%i4) g(x):=cos(x)-sqrt(x);
```

(%o4) $g(x) := \cos(x) - \sqrt{x}$

```
(%i5) wxplot2d(g(x), [x,0,1]);
```

(%t5)



(%o5)

Since `g(0)`> 0 and `g(1)`< 0, there must be a solution in the interval $(0, 1)$.

(If you chose an interval other than $(0, 1)$ then your graph will look different from the one given above. If, in particular, your graph does not cross the x-axis, then you will need to plot the graph of g again using a larger interval.)

```
(%i6) find_root(g(x), x, 0, 1);
(%o6) 0.6417143708728826
```

So a solution of $\cos x = \sqrt{x}$ is $x = 0.642$ (to 3 s.f.).

Solution to Computer activity 28

(a)

```
(%i1) diff(cos(x)/(x^2+3), x);
```

(%o1) $-\frac{\sin(x)}{x^2+3} - \frac{2\,x\cos(x)}{(x^2+3)^2}$

So the derivative of $f(x) = \frac{\cos x}{x^2+3}$ is $f'(x) = -\frac{\sin x}{x^2+3} - \frac{2x\cos x}{(x^2+3)^2}$.

(b)

```
(%i2) diff(u^2*sin(u), u, 2);
```

(%o2) $-u^2\sin(u) + 2\sin(u) + 4\,u\cos(u)$

So the second derivative of $g(u) = u^2 \sin u$ is $g''(u) = -u^2 \sin u + 2 \sin u + 4u \cos u$.

(c)

```
(%i3) diff(log(t)/t^2, t, 3);
```

(%o3) $\frac{26}{t^5} - \frac{24\log(t)}{t^5}$

So the third derivative of $h(t) = \frac{\ln t}{t^2}$ is $h'''(t) = \frac{26}{t^5} - \frac{24 \ln t}{t^5}$.

(d)

```
(%i4) integrate(x*sin(x)^2, x);
```

(%o4) $-\frac{2\,x\sin(2\,x) + \cos(2\,x) - 2\,x^2}{8}$

So $\int x \sin^2 x \,\mathrm{d}x = -\frac{2x\sin(2x) + \cos(2x) - 2x^2}{8} + c$.

(e)

```
(%i5) integrate(t^2*%e^t, t, 0, 1);
(%o5) %e - 2
```

So $\int_0^1 t^2 e^t \,\mathrm{d}t = e - 2$.

(f)

```
(%i6) integrate(1/x^2, x, 1, 2);
```

(%o6) $\frac{1}{2}$

So $\int_1^2 \frac{1}{x^2} \,\mathrm{d}x = \frac{1}{2}$.

(g)

```
(%i7) integrate(tan(exp(-x)), x, 0, 1);
```

(%o7) $$\int_0^1 \tan(\%e^{-x})\mathrm{dx}$$

Maxima cannot evaluate $\int_0^1 \tan(e^{-x})\,dx$ exactly.

Solution to Computer activity 29

(a)

```
(%i1) f(x):=x*sqrt(1+4*x^2);
```

(%o1) $$\mathrm{f(x)} := \mathrm{x}\sqrt{1+4\,\mathrm{x}^2}$$

(b)

```
(%i2) df(x):=diff(f(x),x);
```

(%o2) $$\mathrm{df(x)} := \frac{\mathrm{d}}{\mathrm{dx}}\mathrm{f(x)}$$

(c)

```
(%i3) df(3);
diff: second argument must be a variable; found 3
#0: df(x=3)
-- an error. To debug this try: debugmode(true);
```

(d)

```
(%i4) df(x):=''(diff(f(x),x));
```

(%o4) $$\mathrm{df(x)} := \sqrt{4\,\mathrm{x}^2+1}+\frac{4\,\mathrm{x}^2}{\sqrt{4\,\mathrm{x}^2+1}}$$

So $f'(x) = \sqrt{1+4x^2} + \dfrac{4x^2}{\sqrt{1+4x^2}}$.

(e)

```
(%i5) df(3);
```

(%o5) $$\frac{73}{\sqrt{37}}$$

So $f'(3) = \dfrac{73}{\sqrt{37}}$.

(f)

```
(%i6) F(x):=''(integrate(f(x),x));
```

(%o6) $$\mathrm{F(x)} := \frac{(4\,\mathrm{x}^2+1)^{3/2}}{12}$$

(g)

```
(%i7) F(1)-F(0);
```

(%o7) $$\frac{5^{3/2}}{12}-\frac{1}{12}$$

```
(%i8) integrate(f(x),x,0,1)
```

(%o8) $$\frac{5^{3/2}}{12}-\frac{1}{12}$$

So $\int_0^1 f(x)\,dx = F(1) - F(0) = \dfrac{5^{3/2}}{12} - \dfrac{1}{12}$.

Solution to Computer activity 30

(a) (i) `(%i1) kill(n);`

`(%o1)` *done*

(ii) `(%i2) integrate(x^n, x);`

`Is n equal to -1? n;`

`(%o2)` $\frac{x^{n+1}}{n+1}$

(b) (i) `(%i3) assume(notequal(n,-1));`

`(%o3)` $[\mathrm{notequal}(n,-1)]$

(ii) `(%i4) integrate(x^n, x);`

`(%o4)` $\frac{x^{n+1}}{n+1}$

(iii) `(%i5) facts();`

`(%o5)` $[\mathrm{notequal}(n,-1)]$

So Maxima knows $n \neq -1$.

(iv) `(%i6) forget(notequal(n,-1));`

`(%o6)` $[\mathrm{notequal}(n,-1)]$

(v) `(%i7) facts();`

`(%o7)` $[\,]$

No facts are now known.

Solution to Computer activity 31

(a) `(%i1) integrate(log(sin(x^3)), x, 0, 1);`

`(%o1)` $\int_0^1 \log\left(\sin(x^3)\right) dx$

Maxima cannot evaluate this integral exactly. It returns the integral unevaluated.

(b) `(%i2) quad_qags(log(sin(x^3)), x, 0, 1);`

`(%o2)` $[-3.024256571599231, 1.77635683940025\, 10^{-14}, 231, 0]$

(Remember `1.77635683940025` 10^{-14} means $1.776\,356\,839\,400\,25 \times 10^{-14}$.)

So $\int_0^1 \ln(\sin(x^3))\,dx = -3.02$ (to 3 s.f.).

Solution to Computer activity 32

(a) `(%i1) quotient(59,8);`

`(%o1)` 7

The quotient on dividing 59 by 8 is 7.

(b) `(%i2) remainder(59,8);`

`(%o2)` 3

The remainder on dividing 59 by 8 is 3.

(c)

```
(%i3) quotient(3^100, 2^100+11);
(%o3) 406561177535215237
(%i4) remainder(3^100, 2^100+11);
(%o4) 503611859755855824366132007889
```

So $3^{100} = 406\,561\,177\,535\,215\,237 \times (2^{100} + 11) + 503\,611\,859\,755\,855\,824\,366\,132\,007\,889$.

(d)

```
(%i5) gcd(108,93);
(%o5) 3
```

So the highest common factor (greatest common divisor) of 108 and 93 is 3.

(e)

```
(%i6) gcd(3^100, 2^100+11);
(%o6) 9
```

So the highest common factor (greatest common divisor) of 3^{100} and $2^{100} + 11$ is 9.

Solution to Computer activity 33

(a)

```
(%i1) gcdex(108,93);
(%o1) [−6,7,3]
```

So the highest common factor of 108 and 93 is 3, and $108 \times (-6) + 93 \times 7 = 3$.

(b)

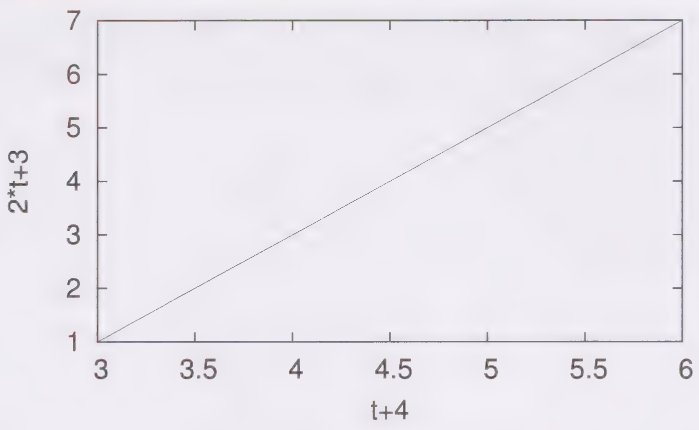
```
(%i2) gcdex(3^100, 2^100+11);
(%o2) [27354783488210640033683046227,
       −11121392986187780374611343715203446360889730414,9]
```

So the highest common factor of 3^{100} and $2^{100} + 11$ is 9, and $3^{100} \times 27\,354\,783\,488\,210\,640\,033\,683\,046\,227 + (2^{100} + 11) \times (-11\,121\,392\,986\,187\,780\,374\,611\,343\,715\,203\,446\,360\,889\,730\,414) = 9$.

Solution to Computer activity 34

(a)

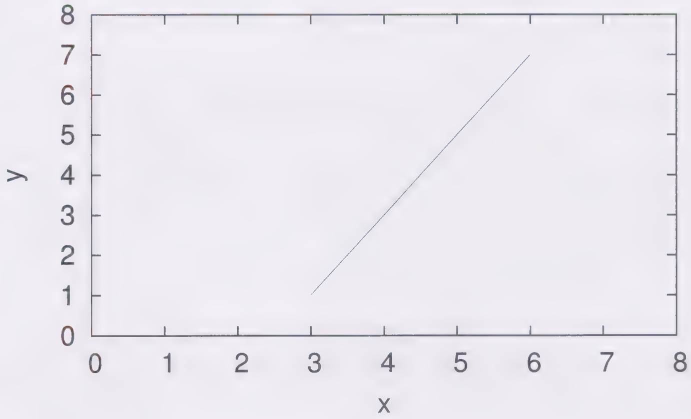
```
(%i1) wxplot2d( [parametric, t+4, 2*t+3, [t,-1,2]] );
(%t1)
```



```
(%o1)
```

(b)

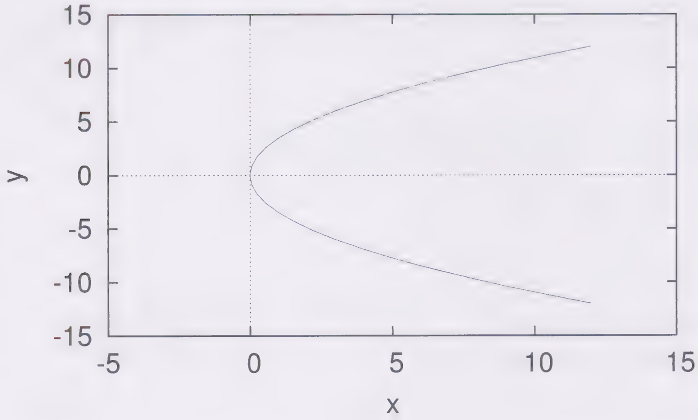
```
(%i2) wxplot2d( [parametric, t+4, 2*t+3, [t,-1,2]],
       [x,0,8], [y,0,8], [xlabel,"x"], [ylabel,"y"] );
(%t2)
```



```
(%o2)
```

(c)

```
(%i3) wxplot2d( [parametric, 3*t^2, 6*t, [t,-2,2]],
      [x,-5,15], [y,-15,15], [xlabel,"x"], [ylabel,"y"]);
(%t3)
```



```
(%o3)
```

Note that axis ranges have been chosen to clearly show the curve represented by the given equations and range of values of t.

Solution to Computer activity 35

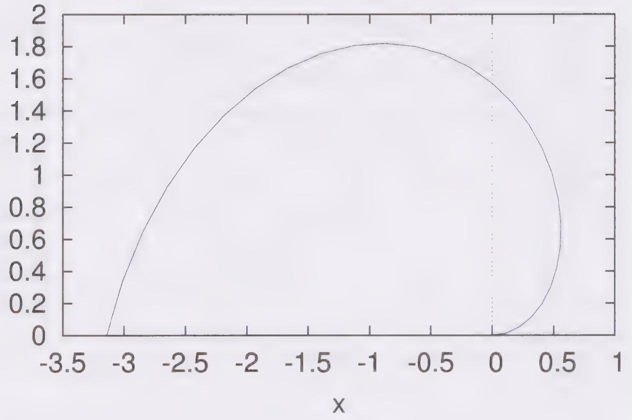
(a)

```
(%i1) x(t):=t*cos(t);
(%o1) x(t) := t cos(t)
(%i2) y(t):=t*sin(t);
(%o2) y(t) := t sin(t)
```

(b)

```
(%i3) wxplot2d( [parametric, x(t), y(t), [t,0,%pi]],
       [xlabel,"x"], [ylabel,"y"] );
```

(%t3)



(%o3)

Solution to Computer activity 36

(a)

```
(%i1) x(t):=12-12*sin(t)^2;
```

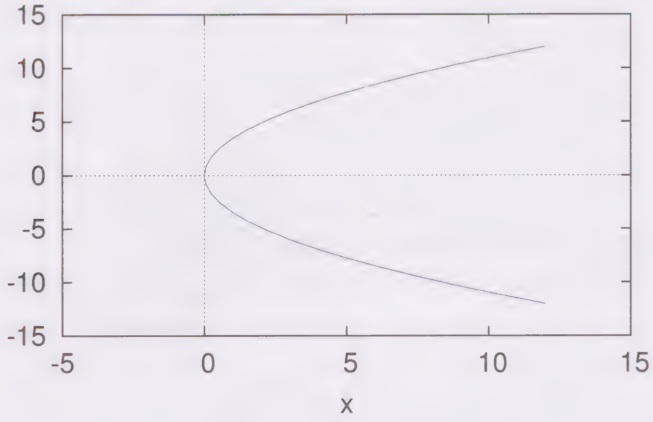
(%o1) $x(t) := 12 - 12\sin(t)^2$

```
(%i2) y(t):=12*cos(t);
```

(%o2) $y(t) := 12\cos(t)$

```
(%i3) wxplot2d( [parametric, x(t), y(t), [t,0,%pi]],
       [x,-5,15], [y,-15,15], [xlabel,"x"], [ylabel,"y"] );
```

(%t3)

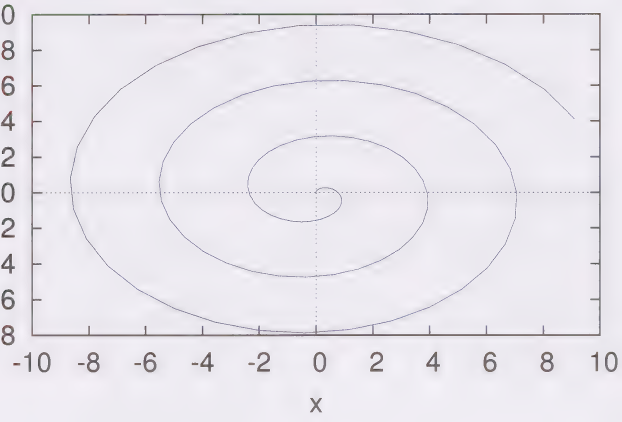


(%o3)

(b)

```
(%i4) wxplot2d( [parametric, t*sin(2*t), t*cos(2*t), [t,0,10]],
       [xlabel,"x"], [ylabel,"y"] );
```

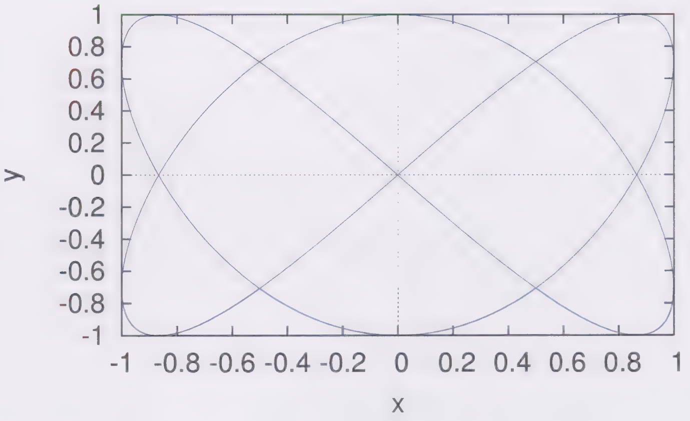
(%t4)



(%o4)

(c) (%i5) wxplot2d([parametric, sin(2*t), cos(3*t), [t,0,20]],
[xlabel,"x"], [ylabel,"y"]);

(%t5)



(%o5)

Solution to Computer activity 37

(a) (i) (%i1) x(t):=3*t^3;

(%o1) $x(t) := 3t^3$

(%i2) y(t):=8*t^2;

(%o2) $y(t) := 8t^2$

(%i3) f(x):=2*x^2;

(%o3) $f(x) := 2x^2$

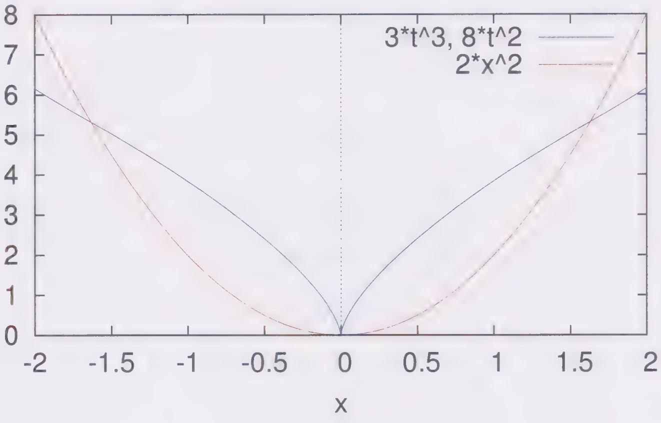
(ii) (%i4) p:[parametric, x(t), y(t), [t,-1,1]];

(%o4) $[parametric, 3\,t^3, 8\,t^2, [t, -1, 1]];$

(iii) (%i5) wxplot2d([p,f(x)], [x,-2,2]);

plot2d: some values were clipped.

(%t5)



(%o5)

(b) (%i6) p:[parametric, sin(t)^3, cos(t)^3, [t,0,2*%pi]];

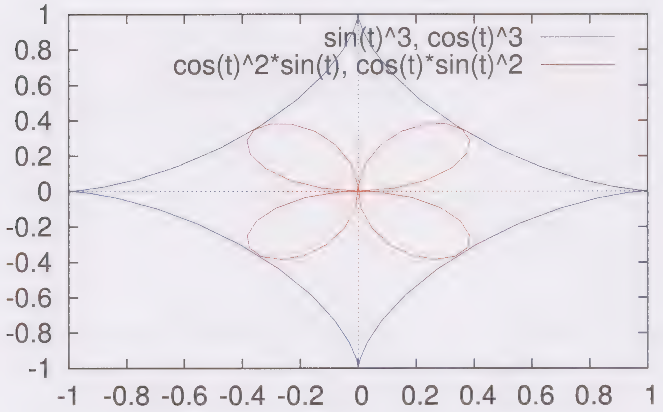
(%o6) $[parametric, \sin(t)^3, \cos(t)^3, [t, 0, 2\pi]];$

(%i7) q:[parametric,cos(t)^2*sin(t), sin(t)^2*cos(t),
[t,0,2*%pi]];

(%o7) $[parametric, \cos(t)^2\sin(t), \cos(t)\sin(t)^2, [t, 0, 2\pi]];$

(%i8) wxplot2d([p, q]);

(%t3)

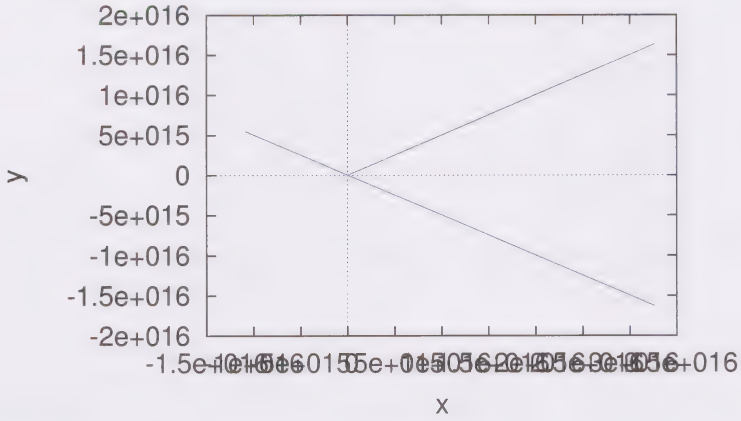


(%o3)

Solution to Computer activity 38

```
(a) (%i1) wxplot2d( [parametric, 3+2*sec(t), -5+tan(t),
           [t,-%pi/2,3*%pi/2]], [xlabel,"x"], [ylabel,"y"] );
```

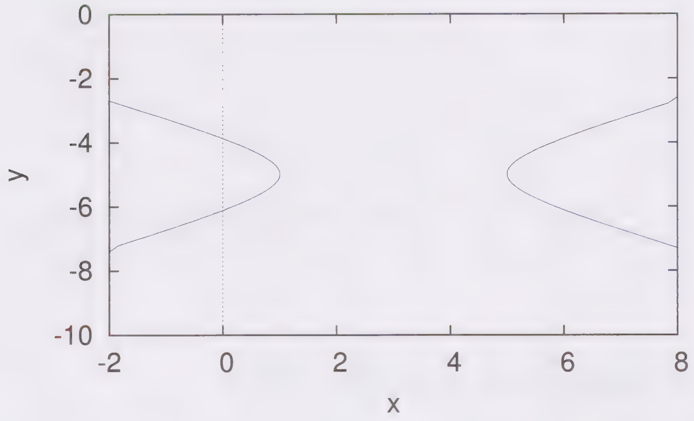
(%t1)



(%o1)

```
(b) (%i2) wxplot2d( [parametric, 3+2*sec(t), -5+tan(t),
           [t,-%pi/2,3*%pi/2]], [x,-2,8], [y,-10,0],
           [xlabel,"x"], [ylabel,"y"] );
```

(%t2)



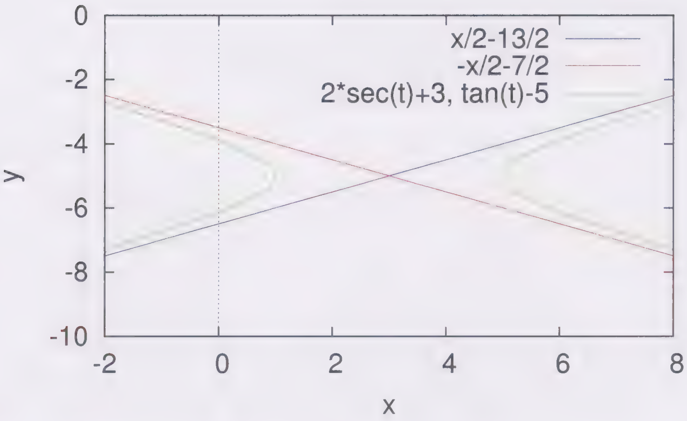
(%o2)

```
(c) (%i3) p:[parametric, 3+2*sec(t), -5+tan(t), [t,-%pi/2,3*%pi/2]];
```

(%o3) $[\mathtt{parametric}, 2\sec(t)+3, \tan(t)-5, [t, -\frac{\pi}{2}, \frac{3\pi}{2}]]$

```
    (%i4) wxplot2d( [x/2-13/2, -x/2-7/2, p], [x,-2,8], [y,-10,0],
           [xlabel,"x"], [ylabel,"y"] );
```

(%t4)

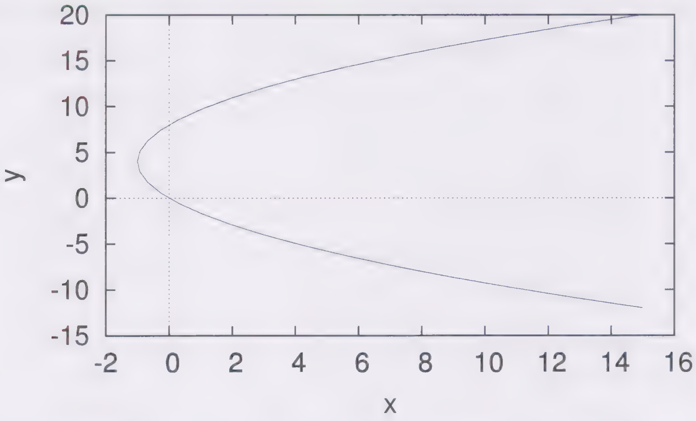


(%o4)

(d)

```
(%i5) wxplot2d( [parametric, -1+4*t^2, 4+8*t, [t,-2,2]],
       [xlabel,"x"], [ylabel,"y"] );
```

(%t5)

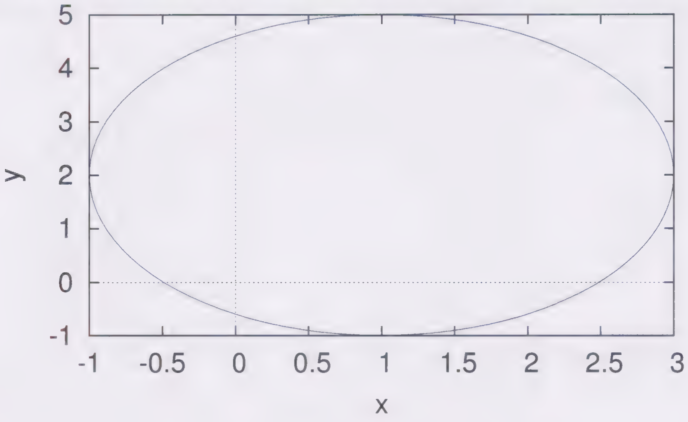


(%o5)

(e)

```
(%i6) wxplot2d( [parametric, 1+2*cos(t), 2+3*sin(t),
       [t,0,2*%pi]], [xlabel,"x"], [ylabel,"y"] );
```

(%t6)



(%o6)

Solution to Computer activity 39

(a)

```
(%i1) partfrac( 7*x/((x+5)*(x-2)), x );
```

(%o1) $\dfrac{5}{x+5}+\dfrac{2}{x-2}$

So $\dfrac{7x}{(x+5)(x-2)}=\dfrac{5}{x+5}+\dfrac{2}{x-2}$.

(b)
```
(%i2) factor(%);
```

$$(\%o2)\quad \frac{7x}{(x-2)(x+5)}$$

Solution to Computer activity 40

(a)
```
(%i1) partfrac( 9/((2*x+1)*(x-1)), x );
```

$$(\%o1)\quad \frac{3}{x-1}-\frac{6}{2x+1}$$

So $\dfrac{9}{(2x+1)(x-1)}=\dfrac{3}{x-1}-\dfrac{6}{2x+1}$.

(b)
```
(%i2) partfrac( 3*x/(x^2+x-2), x );
```

$$(\%o2)\quad \frac{2}{x+2}+\frac{1}{x-1}$$

So $\dfrac{3x}{x^2+x-2}=\dfrac{2}{x+2}+\dfrac{1}{x-1}$.

(Note that the denominator, x^2+x-2, of the given expression can be factorised as $(x+2)(x-1)$.)

(c)
```
(%i3) partfrac( (4*x+8)/(4*x-3)^2, x );
```

$$(\%o3)\quad \frac{1}{4x-3}+\frac{11}{(4x-3)^2}$$

So $\dfrac{4x+8}{(4x-3)^2}=\dfrac{1}{4x-3}+\dfrac{11}{(4x-3)^2}$.

(d)
```
(%i4) partfrac( (8*x+4)/((5*x-3)*(x-1)^2), x );
```

$$(\%o4)\quad \frac{55}{5x-3}-\frac{11}{x-1}+\frac{6}{(x-1)^2}$$

So $\dfrac{8x+4}{(5x-3)(x-1)^2}=\dfrac{55}{5x-3}-\dfrac{11}{x-1}+\dfrac{6}{(x-1)^2}$.

(e)
```
(%i5) partfrac( 26/((x+5)*(x^2+1)), x );
```

$$(\%o5)\quad \frac{5-x}{x^2+1}+\frac{1}{x+5}$$

So $\dfrac{26}{(x+5)(x^2+1)}=\dfrac{5-x}{x^2+1}+\dfrac{1}{x+5}$.

In each case, the denominators of the partial fractions are the linear and quadratic factors of the denominator of the original expression, except that where the denominator of the original expression has a repeated factor, there is an additional partial fraction whose denominator is the square of this factor.

You will learn more details about this in Unit 7.

Solution to Computer activity 41

(a)

```
(%i1) divide(x^5+x+1, x^3+2);
```

(%o1) $[x^2, -2x^2 + x + 1]$

So the quotient is x^2 and the remainder is $-2x^2 + x + 1$.

(b) (i)

```
(%i2) divide(3*x^4-7*x, x-1);
```

(%o2) $[3x^3 + 3x^2 + 3x - 4, -4]$

So the quotient is $3x^3 + 3x^2 + 3x - 4$ and the remainder is -4.

(ii)

```
(%i3) divide(x^6+x^5, 2*x^2+1);
```

(%o3) $\left[\frac{4x^4 + 4x^3 - 2x^2 - 2x + 1}{8}, \frac{2x - 1}{8}\right]$

So the quotient is $\dfrac{4x^4 + 4x^3 - 2x^2 - 2x + 1}{8}$, which could be written $\dfrac{x^4}{2} + \dfrac{x^3}{2} - \dfrac{x^2}{4} - \dfrac{x}{4} + \dfrac{1}{8}$ and the remainder is $\dfrac{2x - 1}{8}$.

Solution to Computer activity 42

(a) (i)

```
(%i1) eqn:'diff(y,x)-y=exp(x)*sin(x);
```

(%o1) $\frac{d}{d\,x}y - y = \%e^{x}\sin(x)$

(ii)

```
(%i2) ode2(eqn, y, x);
```

(%o2) $y = \%e^{x}(\%c - \cos(x))$

So the general solution of the differential equation is $y = e^x\,(c - \cos(x))$, where c is an arbitrary constant.

(b)

```
(%i3) ode2(3*'diff(y,x)+y=x*y^2, y, x);
```

(%o3) $$y = \frac{\%e^{-\frac{x}{3}}}{\%c - \frac{(-3x - 9)\,\%e^{-\frac{x}{3}}}{3}}$$

```
(%i4) fullratsimp(%);
```

(%o4) $$y = \frac{1}{\%c\,\%e^{x/3} + x + 3}$$

So the general solution of the differential equation is $y = \dfrac{1}{ce^{x/3} + x + 3}$, where c is an arbitrary constant.

(c)

```
(%i5) ode2('diff(y,x)=exp(cos(x))-1, y, x);
```

(%o5) $$y = \int \%e^{\cos(x)}\,dx - x + \%c$$

So the general solution of the equation is

$$y = \int e^{\cos x}\,dx - x + c,$$

where c is an arbitrary constant.

Here, Maxima has calculated the solution in terms of an integral that cannot be evaluated analytically.

(d)
```
(%i6) ode2('diff(y,t)=t*y^3-1, y, t);
(%o6) false
```

The ode2 command failed to find an analytic solution to this equation.

Solution to Computer activity 43

(a)
```
(%i1) eqn:t*'diff(y,t)+2*y=t^2;
```

$$\text{(\%o1)}\quad t\left(\frac{d}{dt}y\right)+2y=t^2$$

(b)
```
(%i2) sol:ode2(eqn, y, t);
```

$$\text{(\%o2)}\quad y=\frac{\frac{t^4}{4}+\%c}{t^2}$$

So the general solution is $y=\dfrac{\frac{t^4}{4}+c}{t^2}$, where c is an arbitrary constant.

This can also be written as $y=\dfrac{t^2}{4}+\dfrac{c}{t^2}$.

(c)
```
(%i3) ic1(sol, y=1, t=1);
```

$$\text{(\%o3)}\quad y=\frac{t^4+3}{4\,t^2}$$

So the required particular solution is $y=\dfrac{t^4+3}{4t^2}$.

Solution to Computer activity 44

(a) First find the general solution of the equation.

```
(%i1) eqn:'diff(y,x)=y-(x-2)^2;
```

$$\text{(\%o1)}\quad \frac{d}{dx}y=y-(x-2)^2$$

```
(%i2) sol:ode2(eqn, y, x);
```

$$\text{(\%o2)}\quad y=\left(-(-x^2-2x-2)\,\%e^{-x}+4\,(-x-1)\,\%e^{-x}+4\,\%e^{-x}+\%c\right)\%e^{x}$$

Then find the particular solution required.

```
(%i3) p:ic1(sol, y=1.9, x=0);
  rat: replaced -0.1 by -1/10 = -0.1
```

$$\text{(\%o3)}\quad y=-\frac{\%e^{x}-10x^2+20x-20}{10}$$

This solution can be simplified using the `expand` command.

```
(%i4) p2:expand(p);
```

$$\text{(\%o4)}\quad y=-\frac{\%e^{x}}{10}+x^2-2x+2$$

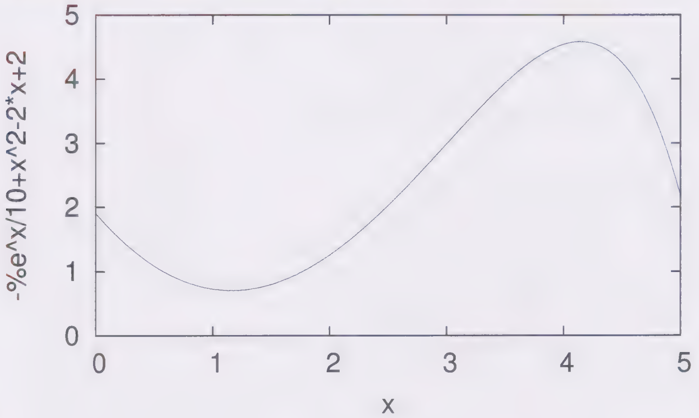
So the required particular solution is

$$y=-\frac{e^x}{10}+x^2-2x+2.$$

(b) The expression to plot is the right-hand side of the equation represented by the variable p2.

```
(%i5) wxplot2d(rhs(p2), [x,0,5], [y,0,5]);
```

(%t5)



(%o5)

Here, a suitable range for y has been chosen.

(c) To find the value of the solution when $x = 4$, substitute 4 for `x` in the right-hand side of `p2`

```
(%i6) subst(4,x,rhs(p2));
```

(%o6) $10-\frac{\%e^4}{10}$

So when $x = 4$, $y = 10 - \frac{e^4}{10}$.

Solution to Computer activity 45

(a)

```
(%i1) load(drawdf);
```

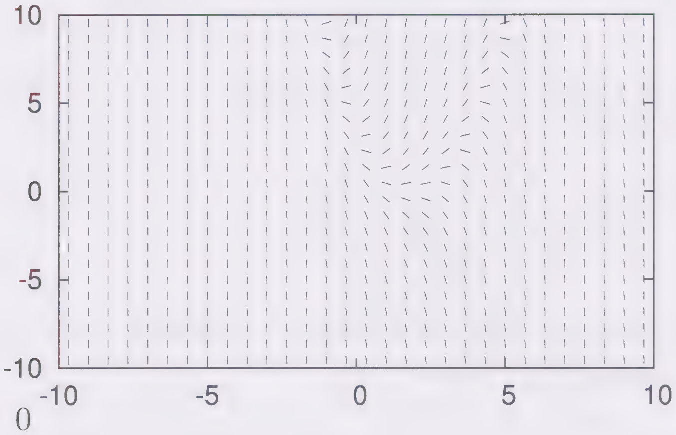
(Some output may appear here.)

```
(%o1) C:\maxima-5.38.1\share\maxima\5.38.1_5\share\
       diffequations\drawdf.mac
```

(b)

```
(%i2) wxdrawdf(y-(x-2)^2, [x,y], field_arrows=false);
```

(%t3)

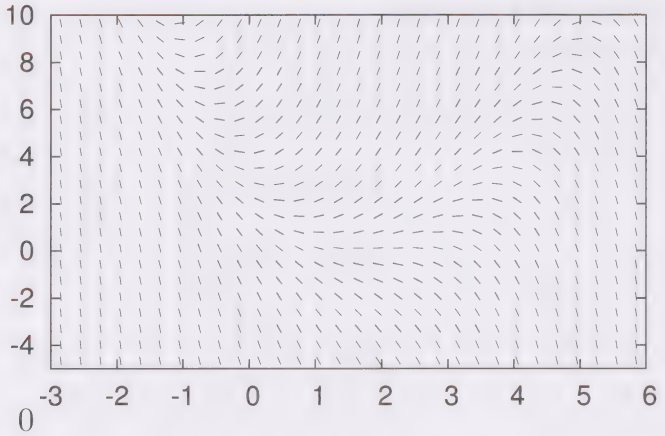


(%o3) 0

(c)

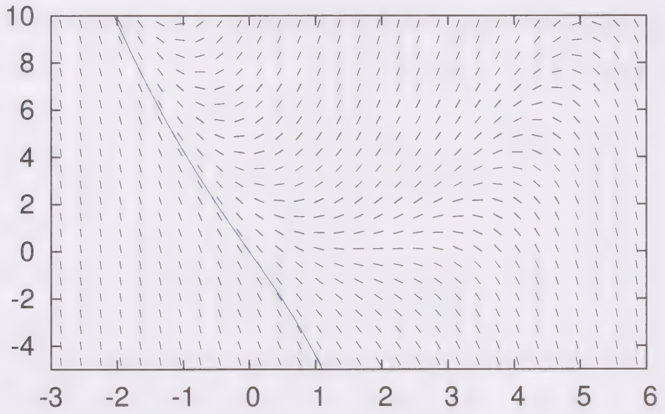
```
(%i3) wxdrawdf(y-(x-2)^2, [x,-3,6], [y,-5,10], field_arrows=false);
```

```
(%t3)
```

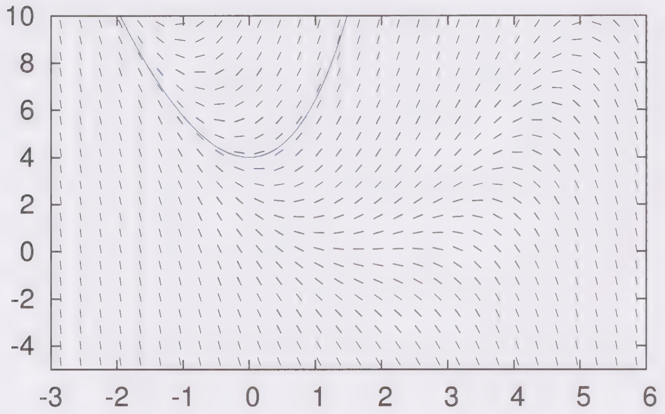


```
(%o3) 0
```

(d) (i) From the direction field, the particular solution passing through the point $(0,0)$ seems to be as shown below.



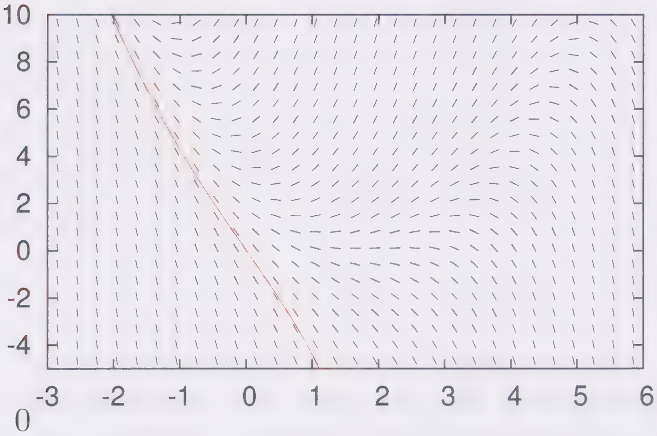
(ii) From the direction field, the particular solution passing through the point $(0,4)$ seems to be as shown below.



(e)

```
(%i4) wxdrawdf(y-(x-2)^2, [x,-3,6], [y,-5,10],
       field_arrows=false,
       [trajectory_at, 0,0]);
```

(%t4)

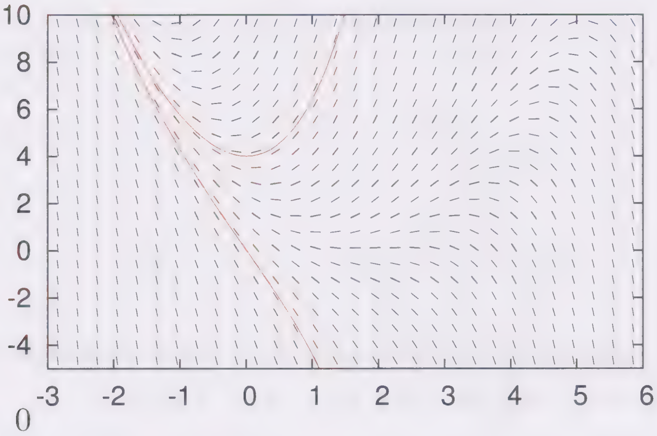


```
(%o4)  0
```

(f)

```
(%i5) wxdrawdf(y-(x-2)^2, [x,-3,6], [y,-5,10],
       field_arrows=false, [trajectory_at, 0,0],
       [trajectory_at, 0,4]);
```

(%t4)

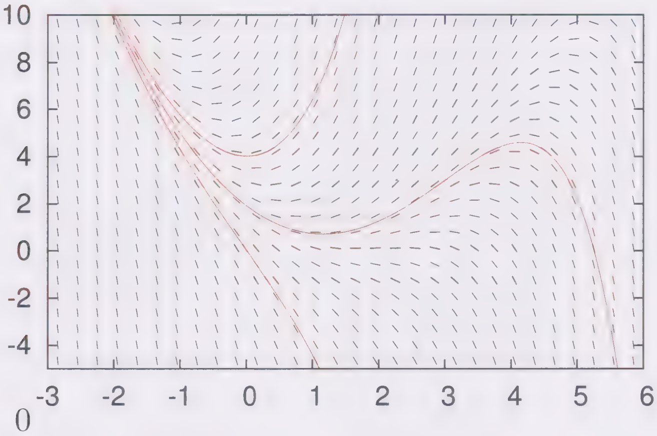


```
(%o4)  0
```

(g)

```
(%i6) wxdrawdf(y-(x-2)^2, [x,-3,6], [y,-5,10],
       field_arrows=false,[trajectory_at, 0,0],
       [trajectory_at, 0,4], [trajectory_at, 0,1.9]);
```

(%t4)



```
(%o4)  0
```

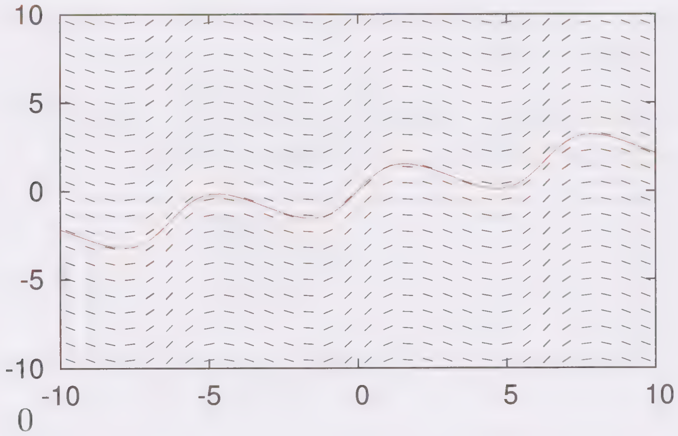
Solution to Computer activity 46

(It is assumed the drawdf package has already been loaded.)

(a) The direction field, together with the approximate particular solution passing through $(0, 0)$, can be plotted as follows.

```
(%i1) wxdrawdf(exp(cos(x))-1, field_arrows=false,
        [trajectory_at, 0,0]);
```

(%t1)



```
(%o1)  0
```

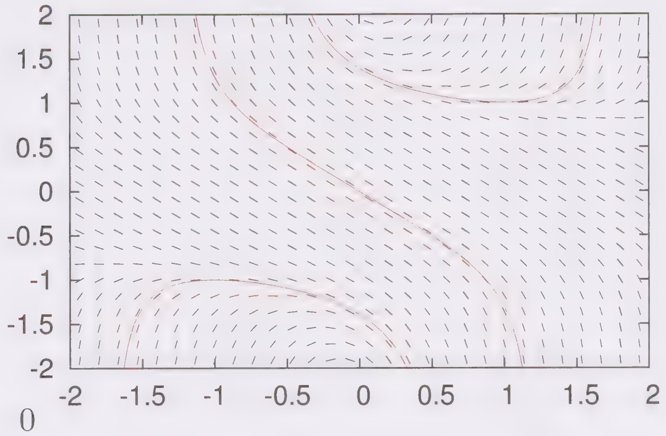
For the particular solution displayed, y generally increases as x increases, but there is some oscillation about this trend.

Every particular solution of this equation will display the same behaviour, since the direction field slopes do not vary as y changes.

(b) The direction field, together with the approximate particular solutions passing through $(0, 0)$, $(1, 1)$ and $(-1, -1)$, can be plotted as follows.

```
(%i2) wxdrawdf(t*y^3-1, [t,-2,2], [y,-2,2], field_arrows=false,
        [trajectory_at, 0,0], [trajectory_at, 1,1],
        [trajectory_at, -1,-1]);
```

(%t2)



```
(%o2)  0
```

For the particular solution that passes through $(0, 0)$, y decreases as t increases, with the gradient of the curve being steeper the larger the magnitude of t.

The particular solution that passes through $(1, 1)$ has a local minimum near $t = 1$. The particular solution that passes through $(-1, -1)$ has a local maximum near $t = -1$.

All the particular solutions of this differential equation will have behaviours similar to one of the above.

Solution to Computer activity 47

(a)

```
(%i1) pts:rk(y-(x-2)^2, y, 1.9, [x, 0, 5, 1]);
(%o1) [[0.0, 1.9], [1.0, 0.70833333333333], [2.0, 1.18923611111111],
      [3.0, 2.783347800925924], [4.0, 3.97573362750771],
      [5.0, 0.66344524116672]]
```

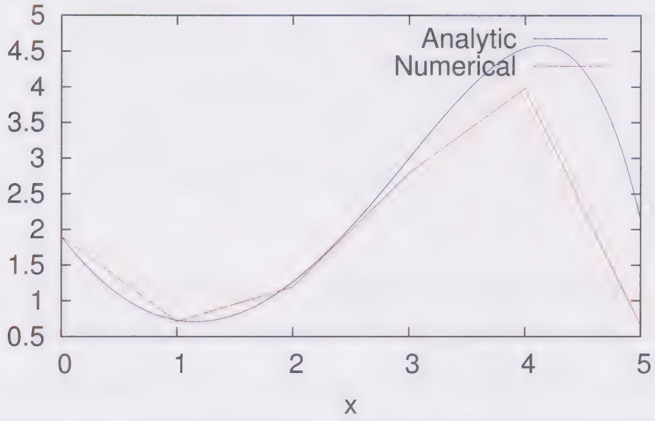
(b)

```
(%i2) asol:-exp(x)/10+x^2-2*x+2;
```

$$(\%o2)\quad -\frac{\%e^{x}}{10}+x^{2}-2x+2$$

```
(%i3) wxplot2d([asol, [discrete, pts]], [x,0,5],
       [legend, "Analytic", "Numerical"]);
```

(%t3)



(%o3)

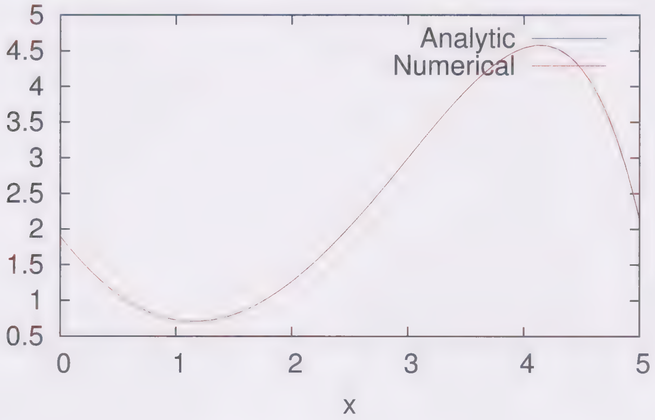
(c)

```
(%i4) pts2:rk(y-(x-2)^2, y, 1.9, [x, 0, 5, 0.1])$
```

(Note that this line has been ended with $ to prevent the display of the calculated points.)

```
(%i5) wxplot2d([asol, [discrete, pts2]], [x,0,5],
       [legend, "Analytic", "Numerical"]);
```

(%t3)

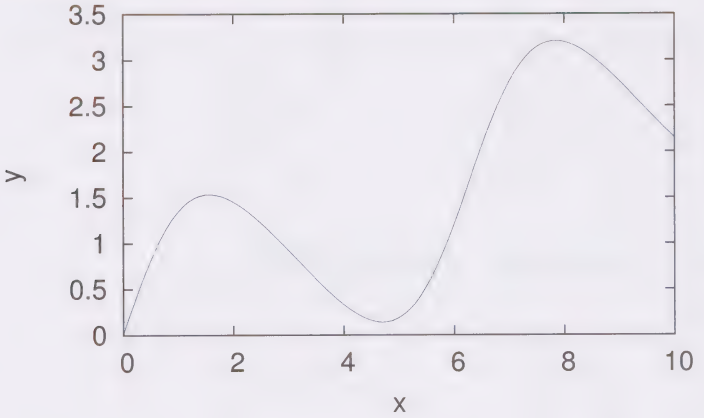


(%o3)

Solution to Computer activity 48

(a)
```
(%i1) pts:rk(exp(cos(x))-1, y, 0, [x,0,10,0.1])$
(%i2) wxplot2d([discrete, pts]);
```

(%t3)



(%o3)

This solution is consistent with the direction field that you found in Computer activity 46(a).

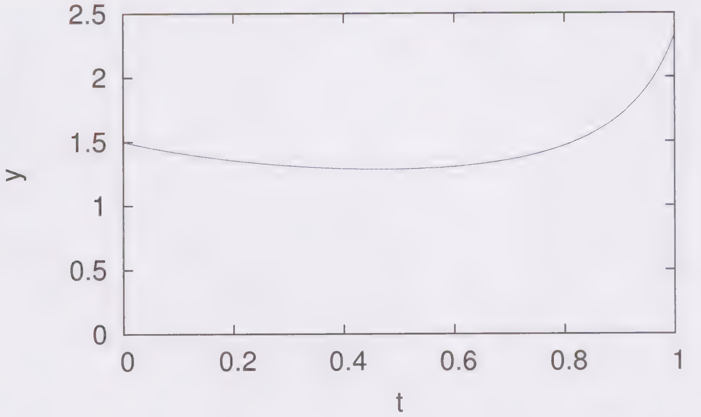
(b)
```
(%i3) pts:rk(t*y^3-1, y, 1.5, [t,0,1,0.01])$
```

Note that here the independent variable is t. A step size of 0.01 has been used to ensure that a smooth curve is obtained.

The approximate solution can be plotted, with suitable axis labels and vertical range, as follows.

```
(%i4) wxplot2d([discrete, pts], [xlabel,"t"], [y,0,2.5]);
```

(%t3)



(%o3)

This solution is consistent with the direction field that you found in Computer activity 46(b).

Solution to Computer activity 49

(a)
```
(%i1) P:matrix( [2,-3,3], [-2,1,2], [1,-1,4] );
```

(%o1) $$\begin{bmatrix} 2 & -3 & 3 \\ -2 & 1 & 2 \\ 1 & -1 & 4 \end{bmatrix}$$

(b) `(%i2) Q:matrix( [5,-3], [-1,-2], [0,4] );`

(%o2) $\begin{bmatrix} 5 & -3 \\ -1 & -2 \\ 0 & 4 \end{bmatrix}$

(c) `(%i3) R:matrix( [1,-2,-1], [4,7,0], [-5,1,3] );`

(%o3) $\begin{bmatrix} 1 & -2 & -1 \\ 4 & 7 & 0 \\ -5 & 1 & 3 \end{bmatrix}$

Solution to Computer activity 50

(a) `(%i4) P+R;`

(%o4) $\begin{bmatrix} 3 & -5 & 2 \\ 2 & 8 & 2 \\ -4 & 0 & 7 \end{bmatrix}$

So $\mathbf{P}+\mathbf{R} = \begin{pmatrix} 3 & -5 & 2 \\ 2 & 8 & 2 \\ -4 & 0 & 7 \end{pmatrix}$.

(b) `(%i5) 3*P;`

(%o5) $\begin{bmatrix} 6 & -9 & 9 \\ -6 & 3 & 6 \\ 3 & -3 & 12 \end{bmatrix}$

So $3\mathbf{P} = \begin{pmatrix} 6 & -9 & 9 \\ -6 & 3 & 6 \\ 3 & -3 & 12 \end{pmatrix}$.

(c) `(%i6) R.Q;`

(%o6) $\begin{bmatrix} 7 & -3 \\ 13 & -26 \\ -26 & 25 \end{bmatrix}$

So $\mathbf{RQ} = \begin{pmatrix} 7 & -3 \\ 13 & -26 \\ -26 & 25 \end{pmatrix}$.

(d) `(%i7) P^^2;`

(%o7) $\begin{bmatrix} 13 & -12 & 12 \\ -4 & 5 & 4 \\ 8 & -8 & 17 \end{bmatrix}$

So $\mathbf{P}^2 = \begin{pmatrix} 13 & -12 & 12 \\ -4 & 5 & 4 \\ 8 & -8 & 17 \end{pmatrix}$.

(e) `(%i8) determinant(P);`

(%o8) -15

So $\det \mathbf{P} = -15$.

(f)

```
(%i9) invert(R);
```

$$(\%o9)\quad \begin{bmatrix} \frac{7}{2} & \frac{5}{6} & \frac{7}{6} \\ -2 & -\frac{1}{3} & -\frac{2}{3} \\ \frac{13}{2} & \frac{3}{2} & \frac{5}{2} \end{bmatrix}$$

So $\mathbf{R}^{-1} = \begin{pmatrix} \frac{7}{2} & \frac{5}{6} & \frac{7}{6} \\ -2 & -\frac{1}{3} & -\frac{2}{3} \\ \frac{13}{2} & \frac{3}{2} & \frac{5}{2} \end{pmatrix}$.

(g)

```
(%i10) x:matrix([0],[1],[1]);
```

$$(\%o10)\quad \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

```
(%i11) P.x;
```

$$(\%o11)\quad \begin{bmatrix} 0 \\ 3 \\ 3 \end{bmatrix}$$

Since $\mathbf{Px} = \begin{pmatrix} 0 \\ 3 \\ 3 \end{pmatrix} = 3\mathbf{x}$, where $\mathbf{x} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$, it follows that $\mathbf{x}$ is an eigenvector of $\mathbf{P}$ and the corresponding eigenvalue is 3.

Solution to Computer activity 51

(a) (i)

```
(%i1) A:matrix([0,1,1],[-1,2,1],[-1,-1,4]);
```

$$(\%o1)\quad \begin{bmatrix} 0 & 1 & 1 \\ -1 & 2 & 1 \\ -1 & -1 & 4 \end{bmatrix}$$

(ii)

```
(%i2) charpoly(A,m);
```

$$(\%o2)\quad -((2-\mathtt{m})(4-\mathtt{m})+1)\mathtt{m}-2\mathtt{m}+6$$

(iii)

```
(%i3) expand(%);
```

$$(\%o3)\quad -\mathtt{m}^3+6\mathtt{m}^2-11\mathtt{m}+6$$

So the characteristic polynomial of $\mathbf{A}$ is $-m^3+6m^2-11m+6$ and the characteristic equation is $-m^3+6m^2-11m+6=0$. This could be expressed in terms of λ as

$$-\lambda^3+6\lambda^2-11\lambda+6=0.$$

(b)

```
(%i4) B:matrix([4,0,0],[3,1,-3],[3,-3,1]);
```

$$(\%o4)\quad \begin{bmatrix} 4 & 0 & 0 \\ 3 & 1 & -3 \\ 3 & -3 & 1 \end{bmatrix}$$

```
(%i5) expand(charpoly(B,m));
```

(%o5) $-\mathtt{m}^3+6\mathtt{m}^2-32$

So the characteristic polynomial of $\mathbf{B}$ is $-m^3+6m^2-32$ and its characteristic equation is $-m^3+6m^2-32=0$. This could be expressed in terms of λ as

$$-\lambda^3+6\lambda^2-32=0.$$

Solution to Computer activity 52

(a)

```
(%i6) eigenvalues(A);
```

(%o6) $[[1,2,3],[1,1,1]]$

So the eigenvalues of $\mathbf{A}$ are 1, 2 and 3, each of which occurs once.

(b)

```
(%i7) eigenvalues(B);
```

(%o7) $[[-2,4],[1,2]]$

So the eigenvalues of $\mathbf{B}$ are -2 (which occurs once) and 4 (which has multiplicity 2).

Solution to Computer activity 53

(a)

```
(%i8) eigenvectors(A);
```

(%o8) $[[[1,2,3],[1,1,1]],[[[1,\frac{1}{2},\frac{1}{2}]],[[1,1,1]],[[1,1,2]]]]$

So the eigenvalues of $\mathbf{A}$ are:

- 1, with corresponding eigenvector $\begin{pmatrix}1\\\frac{1}{2}\\\frac{1}{2}\end{pmatrix}$,
- 2, with corresponding eigenvector $\begin{pmatrix}1\\1\\1\end{pmatrix}$,
- 3, with corresponding eigenvector $\begin{pmatrix}1\\1\\2\end{pmatrix}$.

These are the same as the eigenvectors verified in Activity 9 of Unit 11, except that the eigenvector corresponding to the eigenvalue 1 is half of that given in Activity 9.

(b)

```
(%i9) eigenvectors(B);
```

(%o9) $[[[-2,4],[1,2]],[[[0,1,1]],[[1,0,1],[0,1,-1]]]]$

So the eigenvalues of $\mathbf{B}$ are:

- -2, with corresponding eigenvector $\begin{pmatrix}0\\1\\1\end{pmatrix}$,

- 4 (which has multiplicity 2), with corresponding eigenvectors $\begin{pmatrix}1\\0\\1\end{pmatrix}$ and $\begin{pmatrix}0\\1\\-1\end{pmatrix}$.

Solution to Computer activity 54

(a) (i)

```
(%i1) x[1]:5;
(%o1) 5
(%i2) x[n]:=2*x[n-1]+3;
(%o2) x_n := 2x_{n-1} + 3
```

(ii)

```
(%i3) x[2];
(%o3) 13
```

So $x_2 = 13$.

(iii)

```
(%i4) x[10];
(%o4) 4093
```

So $x_{10} = 4093$.

(b) (i)

```
(%i5) u[0]:4;
(%o5) 4
(%i6) u[1]:9;
(%o6) 9
(%i7) u[n]:=5*u[n-1]-6*u[n-2];
(%o7) u_n := 5u_{n-1} - 6u_{n-2}
```

(ii)

```
(%i8) u[4];
(%o8) 129
(%i9) u[10];
(%o9) 62121
```

So $u_4 = 129$ and $u_{10} = 62\,121$.

Solution to Computer activity 55

(a)

```
(%i1) load(solve_rec);
(%o1) C:\maxima - 5.38.1\share\maxima\5.38.1_5\share\
      solve_rec\solve_rec.mac
```

(b) (i)

```
(%i2) kill(x);
(%o2) done
```

(ii)

```
(%i3) solve_rec(x[n]=2*x[n-1]+3, x[n], x[1]=5);
(%o3) x_n = 2^{n+2} - 3
```

So the closed form for the sequence given by the recurrence system is $x_n = 2^{n+2} - 3$.

(iii)

```
(%i4) kill(u);
```

(%o4) *done*

```
(%i5) solve_rec(u[n]=5*u[n-1]-6*u[n-2], u[n], u[0]=4,
        u[1]=9);
```

(%o5) $\mathrm{u_n} = 3^{\mathrm{n}} + 3\,2^{\mathrm{n}}$

(Note the space between 3 and 2^n, which indicates multiplication. wxMaxima does not display a multiplication sign.) So the closed form for the sequence given by the recurrence system is $u_n = 3^n + 3 \times 2^n$.

Solution to Computer activity 56

(It is assumed the `solve_rec` package has previously been loaded.)

(a)

```
(%i1) solve_rec(x[n]=2*x[n-1]-1, x[n], x[1]=3);
```

(%o1) $\mathrm{x_n} = 2^{\mathrm{n}} + 1$

So the closed form for the sequence given by the recurrence system is $x_n = 2^n + 1$. This is the same as the solution obtained in Activity 20(a) of Unit 12.

(b)

```
(%i2) solve_rec(y[n]=-1/3*y[n-1]+1, y[n], y[1]=12);
```

(%o2) $$\mathrm{y_n} = \frac{3^{2-\mathrm{n}}(-1)^{\mathrm{n}}}{4} + 4\,3^{2-\mathrm{n}}(-1)^{\mathrm{n}-1} + \frac{3}{2^2}$$

This result can be simplified as follows.

```
(%i3) fullratsimp(%);
```

(%o3) $$\mathrm{y_n} = \frac{3^{\mathrm{n}+1} - 135\,(-1)^{\mathrm{n}}}{4\,3^{\mathrm{n}}}$$

So the closed form for the sequence given by the recurrence system is

$$\begin{aligned} y_n &= \frac{3^{n+1} - 135(-1)^n}{4 \times 3^n} \\ &= \frac{3^{n+1}}{4 \times 3^n} - \frac{135}{4} \times \frac{(-1)^n}{3^n} \\ &= \frac{3}{4} + \frac{135}{4 \times 3} \times \frac{(-1)^{n-1}}{3^{n-1}} \\ &= \frac{3}{4} + \frac{45}{4}\left(-\frac{1}{3}\right)^{n-1}. \end{aligned}$$

This is the same as the solution obtained in Activity 20(b) of Unit 12.

(c)

```
(%i4) solve_rec(u[n]=7*u[n-1]-12*u[n-2], u[n], u[0]=2,
        u[1]=7);
```

(%o4) $\mathrm{u_n} = 4^{\mathrm{n}} + 3^{\mathrm{n}}$

So the closed form for the sequence given by the recurrence system is $u_n = 4^n + 3^n$. This is the same as the solution obtained in Activity 30(a) of Unit 12.

(d)

```
(%i5) solve_rec(u[n]=6*u[n-1]-9*u[n-2], u[n], u[0]=2,
       u[1]=7);
```

(%o5) $\mathtt{u_n} = \left(\frac{\mathtt{n}}{3}+2\right) 3^{\mathtt{n}}$

So the closed form for the sequence given by the recurrence system is $u_n = \left(\frac{n}{3}+2\right) \times 3^n$. This is the same as the solution obtained in Activity 30(b) of Unit 12.

Maxima reference guide

Mathematical operations and functions

Operation or function	Syntax	Example
Addition	`+`	`4+3;`
Subtraction	`-`	`4-3;`
Multiplication	`*`	`4*3;`
Division	`/`	`4/3;`
Brackets	`(` *and* `)`	`2*(3+4);`
Powers, for example 2^3	`^`	`2^3;`
	or `**`	`2**3;`
Square root, for example $\sqrt{5}$	`sqrt(■)`	`sqrt(5);`
Exponential, for example e^3	`%e^■`	`%e^3;`
	or `exp(■)`	`exp(3);`
Natural logarithm, ln	`log(■)`	`log(8);`
Magnitude (absolute value)	`abs(■)`	`abs(-3);`
sin	`sin(■)`	`sin(1);`
cos	`cos(■)`	`cos(3*%pi/2);`
tan	`tan(■)`	`tan(%pi/4);`
cosec	`csc(■)`	`csc(2);`
sec	`sec(■)`	`sec(%pi/3);`
cot	`cot(■)`	`cot(%pi/4);`
$\sin^{-1}$	`asin(■)`	`asin(sqrt(3)/2);`
$\cos^{-1}$	`acos(■)`	`acos(1/sqrt(2));`
$\tan^{-1}$	`atan(■)`	`atan(1/2);`
sinh	`sinh(■)`	`sinh(1);`
cosh	`cosh(■)`	`cosh(1);`
tanh	`tanh(■)`	`tanh(1);`
cosech	`csch(■)`	`csch(1);`
sech	`sech(■)`	`sech(0);`
coth	`coth(■)`	`coth(1);`
$\sinh^{-1}$	`asinh(■)`	`asinh(1);`
$\cosh^{-1}$	`acosh(■)`	`acosh(1);`
$\tanh^{-1}$	`atanh(■)`	`atanh(1/2);`

Mathematical operations and functions (continued)

Operation or function	Syntax	Example

Factorial, $n!$	`!`
	or `factorial( )`
		`5!;`
		`factorial(5);`

^nC_r	`binomial(n,r)`
	or `combination(n,r)`
		`binomial(5,2);`
		`combination(5,2);`

^nP_r	`permutation(n,r)` *Note*: `combination` and `permutation` require the `functs` package to be loaded.	`permutation(5,2);`
Equality (within an equation)	`=`	`5=2*x+3;`
Matrix addition	`+`	`P+Q;`
Matrix subtraction	`-`	`P-Q;`
Scalar multiple of a matrix	`*`	`2*P;`
Matrix multiplication	`.`	`P.Q;`
Matrix powers	`^^`	`P^^3;`
Determinant (of a square matrix)	`determinant(matrix)`	`determinant(P);`

Matrix inverse (of a square matrix)	`invert(matrix)`
	or `matrix^^(-1)`
		`invert(P);`
		`P^^(-1);`

Mathematical commands

Operation	Command	Example

Get help on a command	`? command`
	or `describe(command)`
		`? float;`
		`describe(float);`

Find help information whose title contains the given text	`??`
	or `describe( , inexact)`
		`?? flo;`
		`describe(flo, inexact);`

Prevent the evaluation of a command	`'`	`'diff(y,x)`
Force the evaluation of a command	`''( )`	`g(x):=''(diff(x^4,x));`
Load a package	`load(package name)`	`load(implicit_plot);`

Mathematical commands (continued)

Operation	Command	Example
Assign a value to a variable	`variable : value`	`a:23;`
Display the value of a variable	`variable`	`a;`
Define a function	`:=`	`f(x):=2*x+3;`
Evaluate a function at a value	`function(▢)`	`f(1);`

Remove an assigned variable or function	`kill(variable)`
	`kill(function)`
		`kill(a);`
		`kill(f);`

Remove all assigned variables and functions	`kill(all)`	`kill(all);`
Reset all system variables	`reset()`	`reset();`
Convert to a decimal number	`float(▢)`	`float(sqrt(2));`
Expand brackets	`expand(▢)`	`expand((x+1)^2);`
Factorise	`factor(▢)`	`factor(2*x+4*x^2);`
Simplify	`fullratsimp(▢)`	`fullratsimp((2*x+4*x^2)/x);`
Simplify something involving exponentials and logarithms	`radcan(▢)`	`radcan(log(x^2*y));`
Combine logarithms	`logcontract(▢)`	`logcontract(log(a)+log(b));`
Expand trigonometric functions of sums, differences and multiples of angles	`trigexpand(▢)`	`trigexpand(sin(A+B));`
Express powers of sines and cosines in terms of sines and cosines of multiple angles	`trigreduce(▢)`	`trigreduce(sin(x)^2);`
Simplify trigonometric expressions	`trigsimp(▢)`	`trigsimp(sin(x)^2+cos(x)^2);`
Simplify algebraic fractions containing trigonometric functions	`trigrat(▢)`	`trigrat((sin(2*x))/cos(x));`
Partial fraction expansion of a rational expression	`partfrac(expression, variable)`	`partfrac(1/(x^2-1),x);`
Substitute	`subst(value, variable, expression)`	`subst(4,x,x^2+1);` which substitutes 4 for x in x^2+1

Quotient on dividing one number or polynomial expression by another	`quotient(▢,▢)`	`quotient(6,4);`
		`quotient(x^2,x-1);`

Remainder on dividing one number or polynomial expression by another	`remainder(▢,▢)`	`remainder(6,4);`
		`remainder(x^2,x-1);`

The two previous commands combined	`divide(▢,▢)`	`divide(6,4);`
		`divide(x^2,x-1);`

Mathematical commands (continued)

Operation	Command	Example
Highest common factor (greatest common divisor) of two numbers	`gcd(▢,▢)`	`gcd(6,4);`
Bézout's identity: given integers a and b find the highest common factor d and integers v and w such that $av + bw = d$	`gcdex(▢,▢)`	`gcdex(6,4);`
Left-hand side of an equation	`lhs(equation)`	`lhs(4*x+1=2*x-2);`
Right-hand side of an equation	`rhs(equation)`	`rhs(4*x+1=2*x-2);`
Solve an equation exactly	`solve(equation)`	`solve(2*x^2-1=0);`
Solve an equation for a variable, i.e. change the subject	`solve(equation, variable)`	`solve(2*a*b-3*b=0, a);`
Solve simultaneous equations	`solve(list of equations, list of variables)`	`solve([2*x+y=4, x-2*y=1], [x,y]);`
Solve an equation numerically	`find_root(expression, variable, interval start value, interval end value)`	`find_root(2*x^3-1,x,0,1);`
Extract an element of a list	`list[index]`	`A[2];`
Specify a matrix	`matrix(row, row, ...)`	`matrix([1,2],[3,4]);`
Find the characteristic polynomial of a matrix	`charpoly(matrix, variable)`	`charpoly(A,m);`
Find the eigenvalues of a matrix	`eigenvalues(matrix)`	`eigenvalues(A);`
Find the eigenvectors of a matrix	`eigenvectors(matrix)`	`eigenvectors(A);`

Define a recurrence sequence	`sequence[initial term number]: initial value`
	`sequence[n]:= expression containing previous terms`
		`x[1]:1;`
		`x[n]:=2*x[n-1]+x[n-2];`

Calculate a term in a recurrence sequence	`sequence[term number]`	`x[50];`
Solve a recurrence system	`solve_rec(recurrence relation, general term, initial term(s))` *Note*: this command requires the `solve_rec` package to be loaded.	`solve_rec(a[n]=2*a[n-1], a[n],a[1]=1);`

Mathematical commands (continued)

Operation	Command	Example
Differentiate	diff(□ , variable)	diff(sin(x^2),x);
Differentiate multiple times	diff(□ , variable , positive integer)	diff(log(x),x,3); to differentiate three times
Integrate (find an antiderivative)	integrate(□ , variable)	integrate(x^2,x);
Evaluate a definite integral	integrate(□ , variable , lower limit , upper limit)	integrate(sin(x),x,0,1);
Find an approximate value of a definite integral	quad_qags(□ , variable , lower limit , upper limit)	quad_qags(exp(-x^2),x,0,1);
Make an assumption about a variable	assume(□)	assume(a>0);
State two things are equal	equal(□ , □)	assume(equal(n,3));
State two things are not equal	notequal(□ , □)	assume(notequal(n,-1));
Forget a property	forget(property)	forget(a>0);
List all known facts	facts()	facts();
List all known facts about a particular variable	facts(variable)	facts(a);
Solve a differential equation analytically	ode2(equation , dependent variable , independent variable)	ode2('diff(y,x)=x, y, x);
Apply initial conditions to the analytic solution of a differential equation	ic1(solution , variable = value , variable = value)	ic1(sol, y=1, x=0);
Plot the direction field of a differential equation	wxdrawdf(expression , [variable , variable], field_arrows=false, ...) *Note*: this command requires the drawdf package to be loaded.	wxdrawdf(x*y,[x,y], field_arrows=false);
Solve a differential equation numerically	rk(expression , dependent variable , initial value , [independent variable , initial value , final value , step size])	rk(x^2,y,1,[x,0,10,0.1]);

Mathematical commands (continued)

Operation	Command	Example
Plot a graph	`wxplot2d(expression, horizontal range,...)`	`wxplot2d(x^2,[x,0,1]);` which plots the graph of x^2 for x between 0 and 1 `wxplot2d(x^2,[x,0,1],[y,0,2]);` which plots the graph of x^2 for x between 0 and 1, and with vertical axis between 0 and 2
Plot several graphs	`wxplot2d(list of expressions, horizontal range,...)`	`wxplot2d([x^2,2*x],[x,0,1]);` which plots graphs of x^2 and $2x$ for x between 0 and 1
Plot a curve from parametric equations	`wxplot2d([parametric,` expression for x, expression for y, `[parameter,min,max]], ...)`	`wxplot2d([parametric, 2*t, 3*t+1, [t,0,1]]);`
Plot a graph represented by a set of points	`wxplot2d([discrete, list of points])` (use `plot2d` similarly if not using wxMaxima)	`wxplot2d([discrete, [[0,0], [1,1]] ]);`
Plot a curve represented by an equation in implicit form	`wximplicit_plot(equation, horizontal range, vertical range, ...)`	`wximplicit_plot(x^2+2*y^2=1, [x,-1,1], [y,-1,1]);`
Plot several curves represented by equations in implicit form	`wximplicit_plot(list of equations, horizontal range, vertical range, ...)` (use `implicit_plot` similarly if not using wxMaxima) *Note*: the `wximplicit_plot` and `implicit_plot` commands require the `implicit_plot` package to be loaded.	`wximplicit_plot([x^2+2*y^2=1, y=x^3], [x,-1,1], [y,-1,1]);`
Set the default plotting options	`set_plot_option(option)`	`set_plot_option(same_xy);`

Mathematical constants

Constant	Syntax	Example
e	%e	%e^2;
π	%pi	2*%pi;
i	%i	2+3*%i;

Maxima system variables

Variable	Description	Example
fpprintprec	The number of significant figures of a decimal number to display	fpprintprec:3;
functions	The list of all user-defined functions	functions;
values	The list of all user-assigned variables	values;

Maxima graph plotting options

The following options can be added as additional arguments to plotting commands.

Option	Argument	Example
Vertical range	[y, ▒, ▒]	[y, 0, 5]
Curve colour	[color, ▒, ...]	[color, red]
Legend text	[legend, "▒", ...]	[legend, "Car A"]
Turn legend off	[legend, false]	[legend, false]
Position legend at the bottom of the graph	[gnuplot_preamble, "set key bottom"]	[gnuplot_preamble, "set key bottom"]
Position legend to the right of the graph	[gnuplot_preamble, "set key outside"]	[gnuplot_preamble, "set key outside"]
Horizontal axis label	[xlabel, "▒"]	[xlabel, "t (h)"]
Vertical axis label	[ylabel, "▒"]	[ylabel, "s (km)"]
Set axes to have equal scales	[same_xy, true]	[same_xy, true]
Change the size of the graph	wxplot_size:[▒,▒]	wxplot_size:[800,600];

Maxima direction field plotting options

Operation	Argument	Example
Set the range of a variable	[variable,▒,▒]	[x,0,5]
Plot the graph of a particular solution	[trajectory_at, ▒, ▒]	[trajectory_at, 0, 1]
Set the maximum step size to use when approximating a solution	[tstep, ▒]	[tstep, 0.001]
Set the number of steps used when approximating a solution	[nsteps, ▒]	[nsteps, 200]

Maxima packages

Maxima package	Functions added
`drawdf`	Plotting direction fields of differential equations
`functs`	Additional mathematical functions including `combinations` and `permutations`
`implicit_plot`	Plotting curves given by equations in implicit form
`solve_rec`	Solving recurrence systems

wxMaxima keyboard sequences (Microsoft Windows)

Function	Key sequence
Zoom in	`Alt-I`
Zoom out	`Alt-O`
Copy	`Ctrl-C`
Interrupt a calculation	`Ctrl-G`
Re-evaluate the worksheet	`Ctrl-R`
Close wxMaxima	`Ctrl-Q`
Paste	`Ctrl-V`

wxMaxima menu commands (Microsoft Windows)

Operation	Menu name	Menu option
Close wxMaxima	`File`	`Exit`
Save your work	`File`	`Save` or `Save As`
Print the worksheet	`File`	`Print`
Open a worksheet	`File`	`Open`
Configure wxMaxima	`Edit`	`Configure`
Copy	`Edit`	`Copy`
Paste	`Edit`	`Paste`
Zoom in	`View`	`Zoom In`
Zoom out	`View`	`Zoom Out`
Toggle display of main toolbar	`View`	`Main Toolbar`
Re-evaluate worksheet	`Cell`	`Evaluate All Visible Cells`
Insert text cell	`Cell`	`Insert Text cell`
Insert title cell	`Cell`	`Insert Title cell`
Insert section cell	`Cell`	`Insert Section cell`
Insert subsection cell	`Cell`	`Insert Subsection cell`
Interrupt a calculation	`Maxima`	`Interrupt`
Restart Maxima	`Maxima`	`Restart Maxima`
Enter a matrix	`Algebra`	`Enter Matrix...`
Maxima help window	`Help`	`Maxima Help`

Maxima command-line interface commands

Operation	Command
Close Maxima	`quit();`
Interrupt a calculation	`Ctrl-C`
Turn off 2D formatting of maths	`display2d:false;`
Display output left justified	`leftjust:true;`
Start writing a transcript	`writefile("file path");`
Stop writing a transcript	`closefile();`
Playback all the input and output in a session	`playback();`
Playback a range of input and output	`playback([start line number, end line number]);`
Save the state of a session	`save("file path", all);`
Reload a session	`load("file path");`

Acknowledgements

Grateful acknowledgement is made to the following source:

Page 6: Taken from: http://www.ma.utexas.edu/people/in_memoriam/ (Accessed 2014)

Every effort has been made to contact copyright holders. If any have been inadvertently overlooked the publishers will be pleased to make the necessary arrangements at the first opportunity.

Index

abort calculation
  using the command-line interface 96
  using wxMaxima 20
abs 15
absolute value 15
accessibility 90
  changing wxMaxima colours and fonts 91
  using a screen magnifier 92
  using a screenreader 92
  using wxMaxima with keyboard alone 91
acos 15
acosh 142
adding text to a wxMaxima worksheet 27
addition 15
  of matrices 81
annotating a wxMaxima worksheet 27
approximation, numerical
  of a definite integral 57
  of a solution to a differential equation 76
  of a solution to an equation 50
argument of a command 15
asin 15
asinh 142
assign
  a derivative or integral to a function 54
  function 33
  variable 23
assume 56
assumption
  about a variable 56
  equals 56
  forget 56
  list all 56
  not equals 56
atan 15
atanh 142
axis scales, on a graph 43

Bézout's identity 59
binomial 142
binomial coefficients 142
brackets 15
  expanding 35

CAS *see* computer algebra system
cell, in wxMaxima 18
  compress 18
  delete 18
  marker 11, 18
  re-evaluate 21
  section 27
  subsection 27
  text 27
  title 27
changing the size of a graph, in wxMaxima 42
characteristic equation of a matrix 82
characteristic polynomial of a matrix 83
charpoly 83
closefile 99
closing
  command-line interface 95
  wxMaxima 14
colour of graph 40
combination 142
command
  argument of 15
  editing
    using the command-line interface 96
    using wxMaxima 21
  force evaluation of 54
  prevent evaluation of 68
  reusing
    using the command-line interface 96
    using wxMaxima 21
command-line interface 8, 92
  abort calculation 96
  closing 95
  creating transcript 98
  cursor, editing 93
  editing commands 96
  editing cursor 93
  graphs 101
  input line number 93
  interrupt calculation 96
  output line number 94
  plotting graphs 101
  printing 98
  reusing commands 96
  saving work 97
  transcript 98
compress cell, in wxMaxima 18
computer algebra system 5
configure wxMaxima 13, 18

copy, using wxMaxima 22
cos 15
$\cos^{-1}$ 15
cosec 15
cosech 142
cosh 142
$\cosh^{-1}$ 142
cot 15
coth 142
csc 15
csch 142
cursor
- editing
  - in command-line interface 93
  - in wxMaxima 21
- horizontal, in wxMaxima 21

decimal approximation of a number 19
define
- differential equations 67
- functions 33
- recurrence systems 87
- variables 23

definite integral 51
- approximate value of 57

delete
- cell, in wxMaxima 18
- function 35
- recurrence system 88
- variable 25

derivative 51
- assign to a function 54

describe 32
determinant 81
determinant of a matrix 81
diff 51
differential equation 67
- direction field 71
- initial condition 69
- inputting 67
- particular solution of 69
- solving
  - analytically 68
  - numerically 76

differentiate 51
direction field of a differential equation, plotting 71
divide 58, 66
division 15
- of polynomial expressions 66

drawdf
- command 71
- package 71

editing commands
- using the command-line interface 96
- using wxMaxima 21

editing cursor
- using the command-line interface 93
- using wxMaxima 21

eigenvalues 84
eigenvalues of a matrix 84
- multiplicity 84

eigenvectors 85
eigenvectors of a matrix 85
element of a list 48
equal 56
equals
- assumption 56
- within an equation 46

equation 46
- changing the subject of 47
- differential 67
  - direction field 71
  - initial condition 69
  - inputting 67
  - particular solution of 69
  - solving analytically 68
  - solving numerically 76
- left-hand side 48
- parametric, plotting 60
- plotting 38
- rearranging 47
- right-hand side 48
- simultaneous, solving 48
- solving 46
- solving numerically 50

erf 53
evaluate
- an expression 12
- an expression using the command-line interface 94
- force command to 54
- prevent command from evaluating 68
- re-evaluate a cell using wxMaxima 21
- re-evaluate worksheet using wxMaxima 23

- exp 15
- expand 35
- expanding brackets 35
- exponentials 15
- expressions
  - evaluating 12
  - logarithmic and exponential, simplifying 35
  - simplifying 35
  - trigonometric, simplifying 35

- fact *see* assumption
- factor 35
- factorial 142
- factorising 35
- facts 56
- file types, wxMaxima 29
- find_root 49
- float 19
- floating-point number 19
  - displayed precision 26
- force evaluation of a command 54
- forget 56
- forget assumption 56
- fpprintprec 26
- front-ends 7
- fullratsimp 35
- functions 33
  - assign 33
  - define 33
  - delete 35
  - hyperbolic 142
  - inverse hyperbolic 142
  - inverse trigonometric 15
  - plotting 38
  - trigonometric 15
- functions 35

- gamma_incomplete 53
- gcd 58
- gcdex 59
- gnuplot 41
- graphs 38
  - changing default behaviour 44
  - changing line thickness 101
  - changing properties 40, 101
  - colours 40
  - equal scales 43
  - from a set of points 78
  - from equations in implicit form 42
  - from parametric equations 60
  - legend 40
  - of functions 38
  - plotting 38
  - size of, in wxMaxima 42
  - using command-line interface 101
  - vertical range 39
- greatest common divisor 58

- help 31
- highest common factor 58
- horizontal cursor, in wxMaxima 21
- hyperbolic functions 142
  - inverse 142

- ic1 69
- implicit form of an equation, plotting 42
- implicit_plot
  - command 42
  - package 42
- indefinite integral 51
- initial condition, for a differential equation 69
- initial value problem 69
- input line number 12
- input prompt, wxMaxima 11
- inputting
  - differential equations 67
  - matrix 79
  - recurrence systems 87
- installing Maxima 9
- integral
  - assign to a function 54
  - definite 51
    - approximate value of 57
  - indefinite 51
- integrate 51
- interfaces 7
  - command-line 8, 92
  - wxMaxima 7, 10
- interrupt calculation
  - using the command-line interface 96
  - using wxMaxima 20
- inverse hyperbolic functions 142
- inverse of a matrix 81
- inverse trigonometric functions 15
- invert 81

kill
- function 35
- recurrence system 88
- variable 25

Kutta, Martin Wilhelm 76

left-hand side, of an equation 48
legend, of a graph 40
- position 41
- suppress 41

lhs 48
line number
- input 12
- output 12

Lissajous curve 62
list 48
- element of 48
- of assumptions 56

load 100
- package 42
- previous session
  - using the command-line interface 100
  - using wxMaxima 29

loading packages 42
log 15
logarithmic and exponential expressions
- manipulating 35
- simplifying 35

logarithms
- natural 15
- other bases 16

logcontract 35

magnitude 15
manipulating
- expressions 35
- logarithmic and exponential expressions 35
- trigonometric expressions 35

mathematical operations 15
matrix 79
- addition of 81
- characteristic equation of 82
- characteristic polynomial of 83
- determinant of 81
- eigenvalues of 84
- eigenvectors of 85
- inputting 79
- inverse of 81
- multiplication of 81
- multiplicity of eigenvalue 84
- powers of 81
- scalar multiple of 81
- subtraction of 81
- transpose of 81

matrix 79
Maxima 5
- installing 9

menus, wxMaxima 10
multiplication 15
- matrix 81
- scalar and matrix 81

multiplicity of an eigenvalue of a matrix 84

natural logarithm 15
not equals, assumption 56
notequal 56
nsteps 75
numerical approximation
- of a definite integral 57
- of a solution of a differential equation 76
- of a solution of an equation 50

numerical solution
- of a differential equation 76
- of an equation 50

ode2 68
ordinary differential equation *see* differential equation
oudrawdf 71
output line number 12
ouwxdrawdf 71

packages 42
- drawdf 71
- functs 149
- implicit_plot 42
- loading 42
- solve_rec 88

parametric equations
- plotting 60

partfrac 65
partial fraction expansion 65
particular solution, of a differential equation 69
paste, using wxMaxima 22
permutation 142
playback 99

plot2d 38
  using command-line interface 101
plotting 38
  changing default behaviour 44
  changing line thickness 101
  changing properties 40, 101
  colours 40
  direction field 71
  equal scales 43
  equations in implicit form 42
  graphs 38
  graphs of functions 38
  legend 40
  parametric equations 60
  set of points 78
  size of graph, in wxMaxima 42
  using command-line interface 101
  vertical range 39
powers 15
  of a matrix 81
prevent evaluation of a command 68
printing
  using the command-line interface 98
  using wxMaxima 30
prompt, wxMaxima 11

quad_qags 57
quadrature 57
quotient
  on dividing integers 58
  on dividing polynomial expressions 66
quotient 58, 66

radcan 35
re-evaluate
  a cell using wxMaxima 21
  a worksheet using wxMaxima 23
rearranging an equation 47
recurrence systems 86
  changing 88
  defining 87
  deleting 88
  solving 88
remainder
  on dividing integers 58
  on dividing polynomial expressions 66
remainder 58, 66

reset 27
resetting system variables 27
reusing commands
  using the command-line interface 96
  using wxMaxima 21
rhs 48
right-hand side, of an equation 48
rk 76
roots
  higher 17
  square 15
Runge, Carl 76
Runge–Kutta 76

save 100
saving your work
  using the command-line interface 97
  using wxMaxima 29
scalar multiple of a matrix 81
scales, on graph 43
Schelter, William 6
scientific notation 19
screen magnifier, use with wxMaxima 92
screenreader 92
screenshot 30
sec 15
sech 142
section cell, wxMaxima 27
sequences
  given by recurrence systems 86
  term of 88
set_plot_option 44
significant figures 26
simplifying
  expressions 35
  logarithmic and exponential expressions 35
  trigonometric expressions 35
simultaneous equations 48
sin 15
$\sin^{-1}$ 15
sinh 142
$\sinh^{-1}$ 142
size of graph, in wxMaxima 42
solve 46
solve_rec
  command 88
  package 88

solving
  differential equation
    analytically 68
    numerically 76
  equations 46
    numerically 50
    simultaneous 48
  recurrence systems 88
`sqrt` 15
square root 15
subject of equation, changing 47
subsection cell, wxMaxima 27
`subst` 34
substitute 34
subtraction 15
  of matrices 81
symbolic computation 19
syntax 14
system variables 26
  resetting 27

table of contents, wxMaxima 11
`tan` 15
$\tan^{-1}$ 15
`tanh` 142
$\tanh^{-1}$ 142
term of a sequence 88
text cell, wxMaxima 27
title cell, wxMaxima 27
toolbar, wxMaxima 10
`trajectory_at` 75
transcript, creating using the command-line interface 98
`transpose`, applied to a matrix 81
transpose of a matrix 81
`trigexpand` 35
trigonometric expressions
  manipulating 35
  simplifying 35
trigonometric functions 15
  inverse 15
`trigrat` 35
`trigreduce` 35
`trigsimp` 35
`tstep` 75

variable 23
  assign 23
  assumption 56
  define 23
  delete 25
  system 26

worksheet
  wxMaxima 10
    annotate 27
    re-evaluate 23
`writefile` 98
`wxdrawdf` 71
`wximplicit_plot` 42
wxMaxima 7, 10
  abort calculation 20
  adding text 27
  annotating 27
  cell 18
    compress 18
    delete 18
    marker 11, 18
    re-evaluate 21
    section 27
    subsection 27
    text 27
    title 27
  changing colours and fonts 91
  closing 14
  compress cell 18
  configure 13, 18
  copy 22
  cursor
    editing 21
    horizontal 21
  delete cell 18
  editing commands 21
  editing cursor 21
  file types 29
  horizontal cursor 21
  input line number 12
  input prompt 11
  interrupt calculation 20
  line number
    input 12
    output 12
  menus 10
  output line number 12
  paste 22
  printing 30

  - prompt, input 11
  - re-evaluate
    - cell 21
    - worksheet 23
  - reusing commands 21
  - saving work 29
  - table of contents 11
  - toolbar 10
  - use with keyboard alone 91
  - worksheet 10
    - re-evaluate 23
- `wxplot2d` 38
  - for plotting parametric equations 60
- `wxplot_size` 42

BOOK A

Unit 1 Key techniques
Unit 2 Mathematical typesetting
(Provided as a separate item)
Unit 3 Number theory

BOOK B

Unit 4 Conics
Unit 5 Statics
Unit 6 Geometric transformations

BOOK C

Unit 7 Topics in calculus
Unit 8 Differential equations
Unit 9 Mathematical language and proof

BOOK D

Unit 10 Dynamics
Unit 11 Eigenvalues and eigenvectors
Unit 12 Combinatorics

MST125: Essential mathematics 2

Cover images: Designed by Andrew Whitehead, The Open University

SUP 06241 0

The Open University

MST125 Computer algebra guide